THESIS

CLUSTERING SIMILARITY DIGEST BLOOM FILTERS IN SELF-ORGANIZING MAPS

by

John C. Delaroderie

December 2012

Thesis Advisor:	Joel Young
Second Reader:	Craig Martell

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 4–1–2013	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* 2010-09-29—2012-12-14
4. TITLE AND SUBTITLE CLUSTERING SIMILARITY DIGEST BLOOM FILTERS IN SELF-ORGANIZING MAPS		**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) John C. Delaroderie		**5d. PROJECT NUMBER** **5e. TASK NUMBER** **5f. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department of the Navy		**10. SPONSOR/MONITOR'S ACRONYM(S)** **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**
12. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		
13. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.IRB Protocol Number: N/A		

14. ABSTRACT

In response to increasing numbers of cases involving digital media, and the increasing sizes of and number of pieces of media in those cases, forensic investigators are relying increasingly on triage techniques for prioritizing which media to review. This thesis describes a framework for clustering documents aquired during a digital forensics investigation on a self organizing (aka Kahonen) map allowing new documents to be categorized relative to existing documents. Furthermore the presented algorithm avoids the need to work with source documents but with `sdhash` fingerprints allowing a fifty-fold reduction in data required. To test the methodology, document fingerprints are regenerated from the SOM and compared.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON / 19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	71	

NSN 7540-01-280-5500 — **Standard Form 298 (Rev. 8–98)** Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

CLUSTERING SIMILARITY DIGEST BLOOM FILTERS IN SELF-ORGANIZING MAPS

John C. Delaroderie
Civilian, Department of the Navy
B.S., U.S. Naval Academy, 1997

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2012

Author:	John C. Delaroderie
Approved by:	Joel Young Thesis Advisor
	Craig Martell Second Reader
	Peter Denning Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In response to increasing numbers of cases involving digital media, and the increasing sizes of and number of pieces of media in those cases, forensic investigators are relying increasingly on triage techniques for prioritizing which media to review. This thesis describes a framework for clustering documents aquired during a digital forensics investigation on a self organizing (aka Kahonen) map allowing new documents to be categorized relative to existing documents. Furthermore the presented algorithm avoids the need to work with source documents but with `sdhash` fingerprints allowing a fifty-fold reduction in data required. To test the methodology, document fingerprints are regenerated from the SOM and compared.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ANN	Artificial Neural Network
BF	Bloom Filter
NPS	Naval Postgraduate School
SDBF	Similarity Digest Bloom Filter
SDHASH	Similarity Digest Hash
SOM	Self-Organizing Map
USG	United States Government

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

In many ways, the real story I would like to tell is not about this thesis, but the story behind writing it. I am gratefully in debt to so many people for their support, but would like to give special acknowledgement to the following:

My thesis advisor, Joel Young. As my thesis advisor, you were a mentor when I needed direction, a friend when I needed help, and a task master when I just needed to sit down and work. Thank you for your patience, understanding, and encouragement throughout this entire process. I know we faced challenges working on opposite coasts, but I could not have completed my Master's Degree without you. I am forever grateful for your help and cannot thank you enough for everything you have done for me.

Craig Martell for his mentorship and guidance in the computer science program. While you rained on many of my wistful preconceptions about computer science, you guided me to the truth and showed me the beauty behind the numbers.

Cynthia Irvine for providing me an opportunity to attend the Naval Postgraduate School through the Scholarship for Service program. You made a lifelong dream possible for me and so may other SFS students. Thank you for your dedication to the program and the opportunities you have brought us.

My wife, Karen. I love you. Thank you for your unwavering love, support, and encouragement. I never dreamed following you to Monterey would result in a Master's Degree from the Naval Postgraduate School. None of this would have been possible without you. Thank you for reminding me of what's important in life.

Little Raigen. I'm not sure what the future holds, but your smile lights up my heart and I love you.

And finally, my daughter, Elizabeth Ann Delaroderie. Although I didn't know you very long, I will love you forever.

This material is based upon work supported by the National Science Foundation, under grant No. DUE-0414012. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1: Introduction

In the late 1980s, IBM's 3390 Model 1 direct access storage device boasted a 3.78 gigabyte storage capacity and cost upwards of $90,000 [1]. Most smart phones today easily match, if not exceed, this capacity with solid state drives at a fraction of the cost. The increasing storage capacity available to consumers coupled with the falling prices in technology pose a troubling challenge to digital forensic investigators due to the sheer volume of data that must be processed. To meet this challenge, digital forensic investigators have turned increasingly to digital forensic triage to prioritize their work [2], [3], [4], [3]. Digital forensic triage is a term that borrows from the practice of sorting injured patients into categories based on the probability of benefit from immediate medical attention. Digital forensic triage in this case means focusing the investigator's attention into areas likely to yield significant results, where significant results is measured entirely within the context of the investigation.

Digital forensic triage utilizes various tools and techniques to suppress negligible information or expose contextually important data to the investigator. For example:

- Cryptographic hashes can be used to verify the presence of specific files in a media collection, but they are not resilient to any modifications made to a file.
- `bulk_extractor` can quickly extract various artifacts in digital media, such as email addresses, credit card numbers, and website URLs, without using the native file system to identify the data [5].
- `sdhash` can be used to reveal non-trivial similarity between files, suggesting one is a copy, version, or derivative work of the other [2].

In all cases, the digital forensic investigator must know what is of interest in their investigation in order to configure their tool sets properly. Without this prior knowledge, the tools are ineffective for digital forensic triage. This poses an additional challenge for the investigation. If an examiner has a library of hundreds or thousand of files of interest, either stored as cryptographic hashes or in `sdhash` files, then every single file in a newly acquired media collection must be compared to every file in the library. Some of these artifacts can be discarded as a result of the triage, but a large collection of artifacts may remain. The examiner is then left with the question of how to manage these artifacts. Copies of the data, or copies of entire hard drives, can be made, but data duplication can be time consuming and pose the additional challenge of managing extraordinarily large amounts of storage. Cryptographic hashing and forensic tools such as `sdhash` produce digests for fingerprints of the data, yielding a smaller storage footprint. Unfortunately the original data must still be stored and available for most processing tasks. Additionally, cryptographic hashes can only identify bit-for-bit duplicates of files, and `sdhash` determines only file similarity. They do not convey the content of a file.

This thesis is focused on the increasing the breadth of queries that can be answered using only digests without needing to resort to the original data. In particular we examine the approach of projecting the documents onto a two dimensional surface. Clustering similar files in a two-dimensional space will assist the digital forensics examiner to more rapidly triage newly acquired media collections while at the same time convey contextual meaning to the data. If an examiner knows that certain items of interest tend to cluster together in a particular *hot spot* in the two-dimensional mapping, then the examiner can focus his or her attention in this area first as new data is mapped to the space. Not only will this result in the immediate triage of digital forensic data, but the examiner also has a contextual understanding of what the new data is similar to. Self-organizing maps [6] (SOMs), a form of artificial neural networks, are an ideal candidate for clustering data in this manner as their algorithms are adept at categorizing data based on pre-defined features.

1.1 Research Questions

This thesis proposes that a SOM can be used to capture the similarity information in the `sdhash` digests by working directly with Bloom filters as neurons and incorporating randomized-rounding during training. In addition, we propose a method of testing this hypothesis by creating a SOM framework for testing similarity. A self-organizing map is created with neurons whose feature vectors are of the same dimensionality as the Bloom filters used in the `sdhash` digests. The SOM is then trained on a large collection of documents that have individually been processed by `sdhash`. Once the SOM has been trained, the mapping of the digest Bloom filters to the neurons is tested: Digests are produced for both documents used during the training of the SOM and held-out documents are processed by `sdhash`. The digests' Bloom filters are then matched to the neurons of the SOM they are most most similar to. The feature vectors (which are Bloom filters themselves) of these best-matching neurons are then used to generate new `sdhash` digests for similarity comparison.

1.2 Significant Findings

Using our randomized rounding based strategy for training SOMs with `sdhash` digests, we found that as training cycles for the SOM increases, the similarity between a digest and the digest reconstructed using the best-matching neurons in the SOM increases. This indicates that our adapted SOM formulation *is* capturing some of the similarity properties in the digests. The current results are tentative in that results vary quite a bit between training sets and for each training set and number of training cycles, only one SOM was trained due to limitations in processing time.

1.3 Thesis Structure

This thesis is organized as follows:

- Chapter 2 provides a background on SOMs and how they are used in categorizing data, the forensic tool `sdhash`, and Bloom filters for determining set membership as used by `sdhash`.
- Chapter 3 describes the software framework for building the SOM and the experiments whereby a `sdhash` digest is reconstructed from the feature vectors of the neurons of the SOM. The accuracy of the reconstructed digest will be measured by it's similarity to the source digest.
- Chapter 4 presents the results of the experiments and observations of the clustering of binary data in the SOM.
- Chapter 5 provides a final analysis drawn from the previous chapters and proposes opportunities for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2: Background and Related Work

The discovery of similarity between two items, whether they exist in the physical world or as bits of ones and zeroes, is the process whereby connections are made between objects where no connections existed before. Once an item is identified as being similar to another, we are able to better label and categorize the world around us where no organization existed previously. This thesis attempts to use similarity at a binary level to draw connections between data objects as a means of enabling a digital forensics examiner to triage newly acquired media in a way that not only prioritizes his or her work, but also provides context behind the results of the triage.

This chapter begins by briefly discusses clustering as a entry into the use of artificial neural networks for categorizing information autonomously. From there, we look at a specific type of artificial neural network, the self-organizing map, as a appropriate model to build our discussion around. For the feature vectors used by the neurons of the self-organizing map, we introduce Bloom filters as a lead in to the forensic similarity tool `sdhash`. `sdhash` is an ideal tool for determining similarity between two files, and we will use the mechanics of `sdhash` and it's Bloom filters for designing and implementing our experimental framework.

2.1 Information Clustering for Data Classification

Clustering is the process of organizing data into groups based on similarity in observable patterns in the data. During the organizing process, data is not typically categorized into predefined labels, but rather new and unlabled clusters emerge from the data set [7]. The data is represented by feature vectors from which similarity scores are calculated for clustering. These feature vectors can be as simple or complex as the goal of the clustering process may require. For example, feature vectors comprised of document metadata could be used to cluster groups of documents based on creation time. More complex clustering can be derived from the same documents through the use of more meaningful feature vector selection. For instance, clustering of documents by creation time may not be very useful in organizing a large data corpus, but if keywords are indexed from the documents, then clustering could be based on term or n-gram counts. After such a process, the documents would be grouped together based on the similarity of the words found in the documents, from which the natural conclusion derived from such grouping would be that the documents themselves are of similar topics.

Data classification is the natural outgrowth of data clustering. Once a cluster of similar attributes has emerged, a label can be ascribed to the group, and any new data clustered into the same group will share a similar classification. Techniques for data classification using machine learning algorithms provide automated clustering and labelling of complex feature vectors.

2.2 Artificial Neural Networks for Data Classification

Artificial neural networks are one type of machine learning algorithms that can be broadly classified into two categories based on how they "learn" or are "trained" to process information. This process is either supervised with an outside agent guiding the learning process, or the process is unsupervised and the machine learns according to it's own algorithms. Artificial neural networks are often described based on their training methodology, i.e. they are either supervised or unsupervised.

2.2.1 Supervised Learning

Two of the more well-known artificial neural networks implement supervised training in their "learning" process. They are feedforward and feedback neural networks.

Feedforward artificial neural networks are characterized by input neurons feeding information in *one direction* to output neurons. It is possible, but not required, for a feedforward neural network to also have a layer (or multiple layers) of *hidden* neurons between the input and output neuron layers. Each of the neurons is assigned a weight value. For any stimulus, or input pattern, applied to an input neuron, the weights of the connections between the neuron layers determine which output neuron the pattern is mapped to. This mapping is the process of categorizing the input pattern to the patterns known to the artificial neural network.

Before a feedforward neural network can be used to classify a input pattern, the weights must be trained to produce the correct output for *known* input patterns. Because this form of neural network requires supervised training, an outside agent must adjust the weights of the neurons to produce the desired mapping for each input pattern. Provided that the training set was adequately thorough enough, a feedforward neural network can then be used to accurately classify new input patterns that have not been encountered by the neural network before. It can also be expected after training that similar input patterns will also map to the similar output neurons.

Feedback neural networks, also known as recurrent neural networks, are trained and implemented in much the same fashion as feedforward neural networks. An input pattern introduced at an input neuron will move through the neuron layers of a recurrent neural network, but instead of the one-way direction in a feedforward neural network, new connections moving in the reverse direction allow for backwards propagation of signals. This backwards propagation allows for additional computational processing of input patterns at neurons in a variety of different layers. Feedback neural networks also require supervised training in a similar fashion to feedforward neural networks before they can be used to classify new input patterns.

2.2.2 Unsupervised Learning

While both feedforward and recurrent neural networks can be used to classify input patterns, developing the labeled corpus of information to train the artificial neural network on can be a lengthy and expensive process. Unsupervised neural networks do not require a supervised training process to categorize data, but rather use a competitive process to identify and differentiate different input patterns. One such unsupervised neural network is the self-organizing map (SOM), also know as Kohonen maps after the creator. A SOM is an adaptive, unsupervised neural network consisting of a (typically) two dimensional array of neurons that can be used to represent data of much higher dimensionality. As with feedforward and recurrent neural networks, a SOM must be trained first, but unlike the neural networks discussed above, training is largely an automated process. SOM training is a form of manifold learning in which high dimensional data is projected onto a lower dimensional surface.

2.3 Self-Organizing Maps

While earlier examples represented neural networks as two or more layers of input neurons mapped to output neurons, a SOM is much simpler. Each input neuron is directly connected to *each and every* output neuron (Figure 2.1). The output neurons themselves are arranged in a two dimensional array of neurons, commonly referred to as the SOM map, or simply map. Even though the output neurons are characterized as having two dimensions, the output layer itself can be mapped to various three dimensional shapes to take advantage of edge influences in the training process.

Each neuron in the map contains a vector with the same n features as the feature vectors represented by the training data (henceforth referred to as the input vector). The weight vector for neuron i is written as $m_i = [m_{i_1}, m_{i_2}, \ldots, m_{i_n}]$. The weight vectors for each neuron in the map are initialized prior to training the SOM. How the vectors are initialized is one of the parameters that has to be tuned when using SOMs. Sometimes features are initialized to all zero and other times to random values in some range. The training process begins by coarsely adjusting

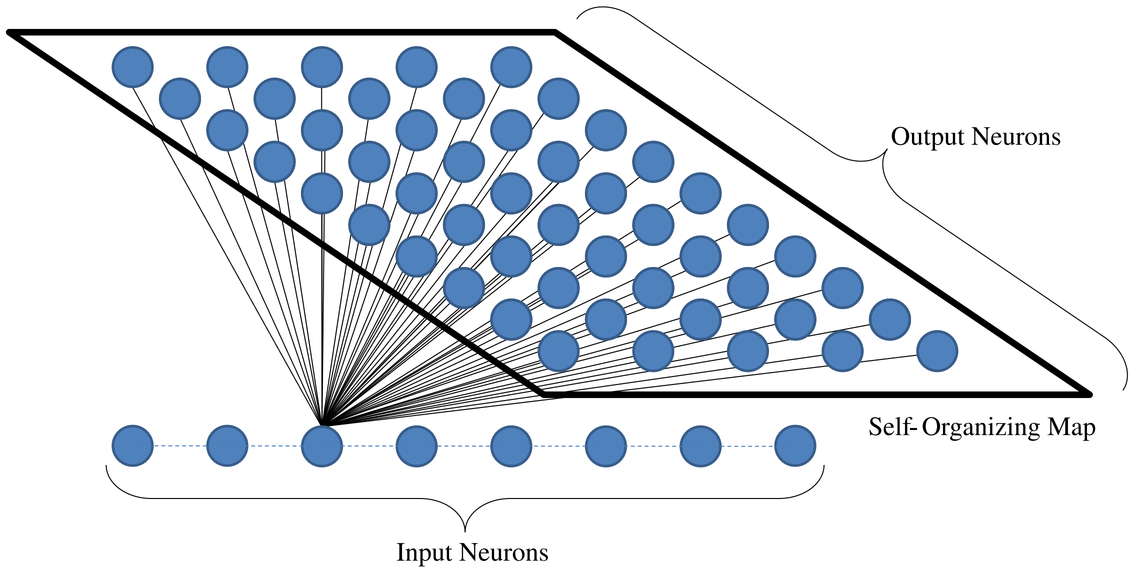


Figure 2.1: In a self-organizing map, each input neuron is mapped to every output neuron

the weight vectors of the map neurons based on a competitive winner selection process that gradually decreases in influence to allow for finer adjustments as training nears completion. In this manner, the map clusters similar neurons of features into groups without the need for user supervision.

The process through which a SOM is trained proceeds in a series of *epochs* composed of smaller steps of *iterations*. Each successive epoch *t* results in finer adjustments to neuron weights. An iteration within an epoch consists of several smaller steps:

- A random input vector is selected from a representative set of training data.
- The output neuron with a feature vector that is most closely similar to the input vector is identified. This is the *best matching neuron*.
- The weight vector of the best matching neuron, along with all the neurons within a certain distance of the best matching neuron (it's *neighborhood*), are adjusted to be more like the input vector. The amount of adjustment is inversely dependent on the distance from the best matching neuron and on the current epoch of SOM training.

While there is no set number of epochs or iterations necessary for training a SOM, the author of [8] recommends that the total number of steps (the product of epochs and iterations) taken during training should be on the scale of 500 times the number of neurons in the SOM. With a conservatively sized 150×150 neuron SOM, the number of training steps is $11,250,000$. The author suggests re-sampling input vectors if necessary in order to perform the recommend number of steps.

For each input vector $x(t)$ randomly selected during a training step t, a "winning" or best matching weight vector $m_{lc_{x(t)}}$ is identified as the SOM neuron with the minimum euclidean distance from the input vector.

$$m_{c_{x(t)}} = \arg\min_i \{ \|x - m_i\| \} \tag{2.1}$$

Once the winning neuron has been determined, the neighborhood of influence $\sigma(t)$ can be found. $\sigma(t)$ consists of all the neurons centered around the winning neuron that must be adjusted to be more similar to the input vector. This adjustment process is what creates clusters of similar neurons within the SOM map. The rate of adjustment is inversely dependent on the distance from the winning neuron. $\sigma(t)$ can initially cover half the size of the SOM and decrease linearly across each successive generation until the radius of σ is the single winning neuron $m_{c_{x(t)}}$. The influence each input vector has on the SOM is calculated by the neighborhood function

$$h_{c(x),i}(t) = \alpha(t) \exp\left(-\frac{\|r_i - r_{c(x)}\|^2}{2\sigma^2(t)} \right) \tag{2.2}$$

where $\alpha(t)$ is a learning rate factor such that $0 < \alpha(t) < 1$. $\alpha(t)$ should remain fixed at a high rate during the initial training to assure proper ordering of the weight vectors. After this ordering has settled, $\alpha(t)$ should decrease substantially to provide finer weight vector adjustments.

The weight vector for each neuron within the radius of σ is adjusted by

$$m_i(t+1) = m_i(t) + h_{c(x),i}(t)[x(t) - m_i(t)] \tag{2.3}$$

After adjusting each neuron within the radius of σ, the next input vector is chosen and the training continues for each iteration until all epochs have elapsed. At this time, the SOM is considered fully trained and all the neurons within the map will have converged into clusters of similar weight vectors [8] [9].

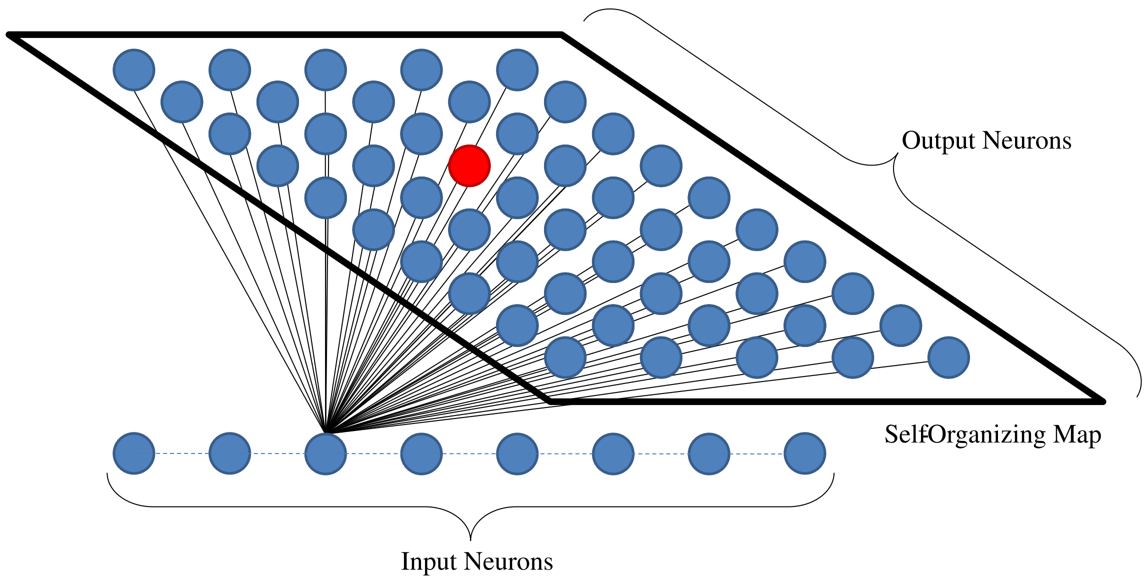


Figure 2.2: An input neuron has identified its best matching neuron in the self-organizing map

Following the training cycle, the SOM can be used to classify new, previously unseen input vectors. The process is similar to the training process: For each input vector selected, the winning neuron is identified. However, no further steps are required - the SOM has categorized the input vector to best match it to the known weigh vectors of the SOM (Figure 2.2).

In applications, once a SOM has been fully trained, it provides a simple visualization for clustering data. Once the features for each neuron are understood, the winning neuron for an input vector provides an immediate analysis of the input stimulus. For example, if a SOM is used to classify a large document archive based on extracted keywords, then for any new document mapped to any given best matching neuron, it can be expected that the new document and the documents clustered in and around the best matching neuron share many similar keywords. As a result, applying SOMs to document archives results in spatially organizing similar documents near one another. Other applications include using citations as feature vectors that result in SOMs that can show clustering of influences for use in technology forecasting [10].

2.4 Feature Selection for SOM Neurons

Keyword-based features can result in SOMs with extremely high dimensionality vectors. Not only does high-dimensionality impact the speed at which a SOM can be trained, but full text-indexing can skew the selection of the winning neuron as well. Frequently used words and

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2.3: Initialized Bloom filter of m bits set to 0

context-poor words (e.g. ‘the’, ‘a’, ‘if’) can be manually or automatically redacted from a dictionary list extracted from the document archives [11]. Another technique is to adjust the weight of keywords based on their term frequency $\times$ inverse document frequency [12].

$$weight_{keyword} = tf_{keyword} \times idf_{keyword} \tag{2.4}$$

where

$$tf_{keyword} = \frac{number\ of\ occurrences\ of\ keyword\ in\ a\ document}{total\ number\ of\ words\ in\ a\ document} \tag{2.5}$$

and

$$idf_{keyword} = \log\left(\frac{total\ number\ of\ documents}{number\ of\ documents\ containing\ keyword}\right) \tag{2.6}$$

These techniques presuppose the use of dictionary words as features. This is sufficient when clustering text based documents with a SOM neural network. However, to use a SOM for clustering data without text based features requires the selection of content appropriate features. For example, file size or creation date are two non-text based features that could be used to cluster files. However, The results of such clustering would have a very limited scope for a forensics examiner. Instead, when clustering binary files by similarity, we propose that a useful feature for the input vectors are the bits in `sdhash` Bloom filters.

2.5 Bloom Filters

A Bloom filter is a data structure used to probabilistically determine set membership. Once a Bloom filter is populated, a simple query returns whether or not an item exists in the filter. While the technique precludes returning false negatives, it is possible for a Bloom filter to return a false positive. The parameters of the Bloom filter can be calculated to yield a particular desired false positive rate for a given number of elements.

To create a Bloom filter for a set with n items, $S = x_1, x_2, x_3, \ldots, x_n$, a size m array of bits is initialized with each bit set to zero (Figure 2.3).

The Bloom filter uses k $log_2 m$-bit hash functions $(h_1, h_2, \ldots, h_k)$ to transform set items into bit

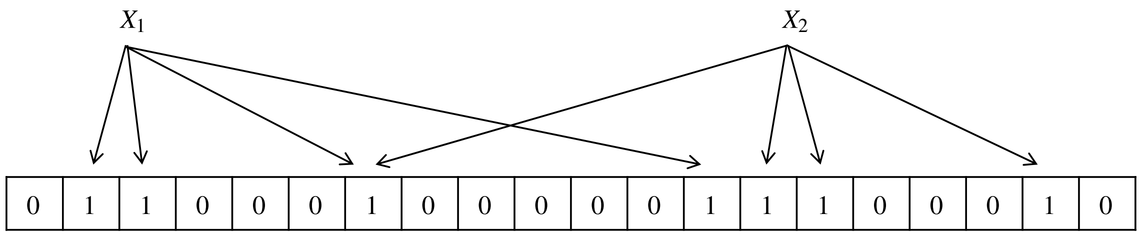


Figure 2.4: The values returned by the hash functions x_1 and x_2 determine the the bit location in the Bloom filter that are set.

locations in the Bloom filter's array. Each element in set S is hashed with each of the k hash functions. The hash functions result in a bit location across the range $1,...,m$. These bit locations in the Bloom filter are set to 1 (Figure 2.4. If a hash function specifies a bit location already set to 1, no change is made — a bit can only be a one or a zero, and once it is toggled to one, it can never be toggled back to zero.

Once each item in the set has been hashed k times, and every bit location returned by the hashes have been set to 1, the Bloom filter can be used for querying set membership. If we want to test if y is a member of S, we simply hash y against each of the k hash functions. Once again, bit locations in the Bloom filer are returned by each of the hashes. This time, however, we do not change any of the bits in the array to 1. Instead, if the bit locations identified by the hashes of y are *all* set to 1 in the Bloom filter, then y *might* be a subset of S. Otherwise, if any of the bit locations are set to 0, then y cannot be in the set S.

It is said that item y *might* be in the set S because Bloom filters can return false positives. This can occur due to chance when different items x in set S coincidentally set all of y's bits to 1.

The probability of a false positive is

$$(1-\rho)^k \approx (1-p')^k \approx (1-p)^k \tag{2.7}$$

where the probability that a bit remains zero after all hash insertions is given by

$$p' = (1-\frac{1}{m})^{kn} \approx e^{-kn/m} \tag{2.8}$$

and p is the asymptotic approximation of p' given by

$$p = e^{-kn/m} \tag{2.9}$$

and ρ is the ratio of 0 bits to total bits remaining in the array after the k hashes have been performed [13].

2.6 SDHASH Bloom Filters

The similarity digest hash (`sdhash`) tool discussed in [14] uses statistically-improbable feature selection for the basis of data fingerprinting. The features are said to be statistically-improbably because they are selected through a process shown to minimize the probability that they will occur by chance in other binary data. Once binary data from a given file has been fingerprinted with `sdhash`, the similarity between two fingerprinted files can easily be calculated. Because `sdhash` uses Bloom filters to fingerprint binary data, the Bloom filters (each a container of feature information) become potentially useful features for use in SOM neural networks.

Every 64 consecutive byte sequence of data in a file represents a candidate feature. For each feature, several scores are calculated in determining which features are statistically-improbable and best represent a unique data fingerprint for the binary file. These features are then inserted to a Bloom filter.

Given a feature B of 64 bytes and a sliding window W of 8 consecutive features:

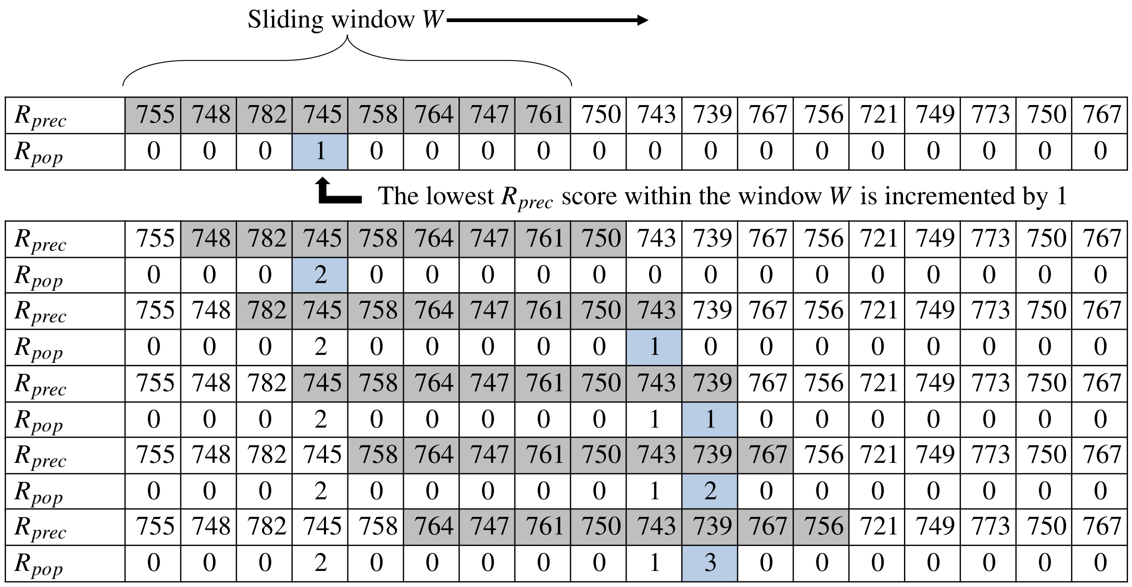
H_{norm} is the entropy score for the feature. After initializing H_{norm} to zero, we first find the Shannon entropy for B. Given the probability $P(X_i)$ of encountering byte i, the Shannon entropy is found where:

$$H = -\Sigma_{i=0}^{255} P(X_i) \log P(X_i) \tag{2.10}$$

and H_{norm} is calculated as:

$$H_{norm} = \lfloor [\rfloor 1000 \times \frac{H}{log_2 W} \rfloor [\rfloor \tag{2.11}$$

R_{prec} is the precedence rank. The H_{norm} entropy scores are ranked in order of least to greatest


Sliding window W
The lowest R_{prec} score within the window W is incremented by 1

R_{prec}	755	748	782	745	758	764	747	761	750	743	739	767	756	721	749	773	750	767
R_{pop}	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

R_{prec}	755	748	782	745	758	764	747	761	750	743	739	767	756	721	749	773	750	767
R_{pop}	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R_{prec}	755	748	782	745	758	764	747	761	750	743	739	767	756	721	749	773	750	767
R_{pop}	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	0	0
R_{prec}	755	748	782	745	758	764	747	761	750	743	739	767	756	721	749	773	750	767
R_{pop}	0	0	0	2	0	0	0	0	0	1	1	0	0	0	0	0	0	0
R_{prec}	755	748	782	745	758	764	747	761	750	743	739	767	756	721	749	773	750	767
R_{pop}	0	0	0	2	0	0	0	0	0	1	2	0	0	0	0	0	0	0
R_{prec}	755	748	782	745	758	764	747	761	750	743	739	767	756	721	749	773	750	767
R_{pop}	0	0	0	2	0	0	0	0	0	1	3	0	0	0	0	0	0	0

Figure 2.5: The leftmost lowest R_{prec} value is identified for each sliding window W. The associated R_{pop} is then incremented.

frequency. The associated ranking for a given H_{norm} entropy score for a particular feature B is then inserted as the R_{prec} score.

R_{pop} is the popularity score. Initially the R_{pop} score is set to zero for each feature. The sliding window W compares the R_{prec} scores across several features. Within this window, the leftmost lowest value R_{prec} is selected. Once selected, the R_{pop} score for that feature is incremented by one 2.5. While most R_{pop} scores will remain zero, some of the scores will increment to higher values. This signifies that the features occur infrequently and are of interest as statistically-improbably features. A minimum threshold value t can be used to further refine the features selected.

Once the statistically-improbably features are selected, those deemed to be weak features must be filtered out. A weak filter is a statistically-improbable feature that unfortunately is not helpful as an identifying feature for data fingerprinting. Through experimentation, the author in [14] filtered features with entropy scores less than 100 and greater than 990. Entropy scores less than 100 are typically found in data with large blocks of identical character sets, such as groups of all zeroes. Entropy scores greater than 990 tended to be high entropy encoding tables shared among many files and, as weak features, not very useful as an identifying feature.

After filtering weak features, the remaining features can be inserted into Bloom filters for creating data fingerprints digests. `sdhash` uses Bloom filters of 256 bytes in length. As before, each element of the Bloom filter is initially set to zero. Generically speaking, once 160 elements have been added to the filter, a new Bloom filter for that file is created. This process repeats until all insertions have been completed. A large data file can result in an equally large number of Bloom filters associated with it in the finished digest.

In creating the digest with `sdhash`, the generic standard mode of the program does not specify block alignment for how a file is split for feature selection. Experimentally it has been shown that the standard mode of sdhash uses between 9 and 10 kilobytes of data on average for each Bloom filter. [15]

There are some disadvantages to the standard mode, and if sdhash instead uses fixed block sizes for each Bloom filter, it may be possible to make faster comparisons (fewer Bloom filters) as well as parallelize the `sdhash` digest creation process. Furthermore, each Bloom filter can be identified with the specific data block from which it was created. The current version of sdhash allows for the creation of block-aligned digests with block sizes of 4K, 8K, and 16K. The use of larger block sizes is not always advantages when applying sdhash to small file sizes. Instead of filling each Bloom filter to a maximum number of features, block mode maps one data block to one Bloom filter. As a result of potentially larger feature sets, the maximum number of features when using block mode is 192. [15]

Regardless of whether block mode or non-block mode is used, each feature of 64 bytes is hashed with SHA-1, yielding a 160 bit hash. The hash is split into 5 equal subhashes of 32 bits each. Eleven bits ($log_2 2048$) from each of the 5 subsections are used as the k hashes for Bloom filter insertions (Figure 2.6).

For each subhash of 32 bits, the 11 least significant bits are used for identifying byte locations in the 256 byte array of the Bloom filter. As before, each Bloom filter location identified can only be set once no matter how many times it is chosen by different hashes [14].

The end result is a series of Bloom filters joined together into a Similarity Digest Bloom Filter (sdbf). The Bloom filters are populated by bits set from hashing statistically-improbably features, and the resulting digest is a unique digital fingerprint for the original data. The Bloom filters themselves can be used for calculating similarity through set intersections.

`sdhash` provides a tool for finding the similarity between two digests. Each Bloom filter in a

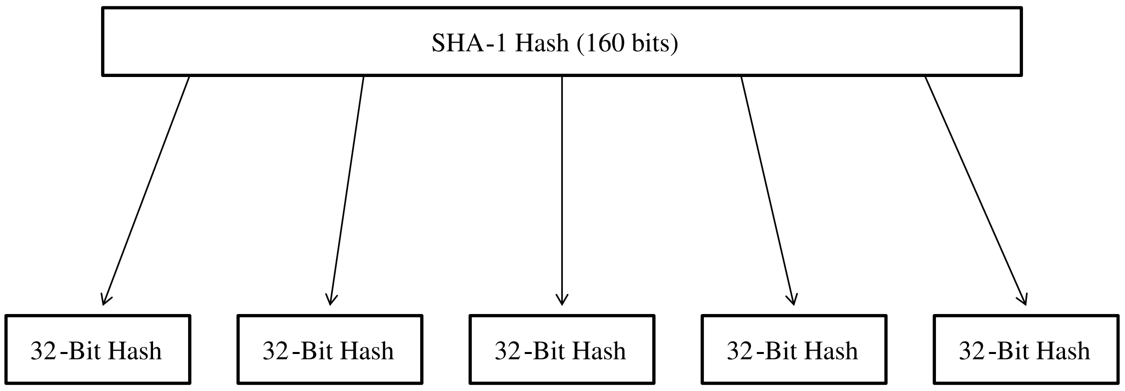


Figure 2.6: A SHA-1 Hash (160-bits) is split into five 32-bit hashes.

given digests is compared to every Bloom filter in the other digest. For example, Bloom filter 1 in digest 1 is compared to all the Bloom filters in digest 2. The maximum similarity score for digest 1 Bloom Filter 1 is recorded. After all of digest 1's Bloom filters have found their maximum similarity scores against all of the Bloom filters in digest 2, the maximum scores are averaged together and a total similarity score between the two digests is created [14].

When calculating similarity, `sdhash` reports a score based on the Bloom filter intersections in the range of 0 to 100. The scores reflect the following similarity: A negative similarity (0), a weak similarity (1 - 10), a marginal similarity (11 - 20), and strong similarity (21 - 100) [2]. Due to the usefulness of `sdhash` in determining file similarity though application of Bloom filter set membership of statistically improbable features, the use of `sdhash` for generating Bloom filters for matching against feature vectors of a SOM neural network are an ideal vector for clustering data at a binary level.

We now have a framework to test our hypothesis on. We have demonstrated how SOMs work and why they are ideally suited as a form of unsupervised learning artificial neural network. `sdhash` creates a digital fingerprint in the form of a digest composed of Bloom filters, and the Bloom filters lend themselves readily as feature vectors to the neurons of our SOM. Finally, `sdhash` itself provides us a means to score similarity between source digests and the digests we create from the feature vectors of the SOM's neuron network. In the next chapter, we will define our testing framework and outline how exactly the SOM generated digests will be created and tested.

CHAPTER 3: Experiment Design and Implementation

In this chapter, we present the experiments designed to test our hypothesis and the infrastructure required for those experiments. In addition, we present our adaptions to the classic SOM formalism needed to work with Bloom filters.

3.1 Experimental Framework

The experimental framework consists of the following components:

- An artificial neural network self-organizing map engine
- A digest generator to construct sdbf digest files from SOM neuron mappings
- `sdhash-2.1` for both the creation of sdbf digests and using the `sdhash` query for file similarity scoring.
- The Naval Postgraduate School government document corpus.

3.1.1 Artificial Neural Network Self-Organizing Map Engine

The first step in creating a SOM is to determine the size of the map in its two dimensions (rows and columns). The number of neurons in the SOM should be greater than the total number of Bloom filters in the document collection used for training the SOM. For instance, if a 1000 document collection yields sdbf digests containing a total of 40,000 Bloom filters, then a square shaped SOM should have rows and columns equal the to square of the total number of Bloom filters, or in this particular example, 200 rows and 200 columns. The neurons are fixed size feature vectors of 2048 bits – the size of a single `sdhash` Bloom filter. During initialization, each bit of the neurons will be randomly set to either a one or zero with an average number of bits set per Bloom filter centered around the expected number of bits ($E[b]$) given the average number of features per filter:

$$E[b] = 2048 - 2048\left(\frac{2047}{2048}\right)^{\hat{f}k} \tag{3.1}$$

Where $\hat{f}$ is the number of features (elements) stored in the filter and k is the number of hashes used by the `sdhash` algorithm (default is 5). $\hat{f}$ is determined by examining the training data set.

Epochs and iterations must also be configured to produce optimal training in the SOM. As discussed in 2.3, the number of steps taken during training, a combination of iterations within epochs, should be on the order of 500 times the number of neurons on the map. Continuing with our previous example, our 40,000 neurons multiplied by 500 would require 20 million steps for training. This can be accomplished with 200 epochs of 100,000 iterations each. Unfortunately, this scale of operation is quite time intensive. For our experiments under time constraints, we unfortunately scaled the training back significantly and instead trained various SOMs with training cycles consisting of 100,000, 1 million, 2 million, and 3 million steps each.

A initial learning rate and decay rate for neighborhood size centered around the best matching neuron must also be defined. These variable can be adjusted experimentally to find a value that results in ideal convergence during training. (There are techniques of training SOMs without these variables, section 5.1 discusses one such method.) All of the aforementioned SOM variables are stored in a separate configuration file which is read into the main program for constructing the SOM neural network during the initialization and training phases. The program takes the SOM configuration file and a collection of sdbf digests as arguments. Training consists of two primary components: a neuron adjustment phase where the neurons of the SOM are adjusted to better resemble the input data, and a final best-matching phase once all the adjustments have been completed.

During training, every Bloom filter in every sdbf digest is stored in a container. During each step of a generation-epoch loop, a Bloom filter is randomly sampled from the container. The entire SOM is iterated through neuron by neuron, and each neuron's feature vector is compared to the randomly selected Bloom filter. As previously defined, the best matching neuron is the neuron whose feature vector most closely resembles the Bloom filter input. This is scored through the use of L2 Norm to find the neuron with the minimum value, whereupon the winning neuron is chosen. Depending on the generation, the learning rate and the neighborhood size of influence will have decayed, and each neuron within the new neighborhood is adjusted based on the decreasing learning rate. Each of the 2048 bits in the feature vector of the neuron can be potentially adjusted. The first bit in the feature vector is compared to the first bit of the sampled Bloom filter. If the two bits match (i.e. both are 1 or both are 0), no change is made as the bit in the feature vector is already similar to the corresponding bit of the sampled Bloom filter. If the two do not match, the feature vector bit is adjusted as described in equation 2.3. Because the feature vector of the neuron is composed of integer values of either 1 or 0, the actual adjustment of the neuron is based on randomized rounding [16]. Randomized rounding allows us to avoid fractional values in the feature vector by changing the adjusted value to a 1 or 0 based on the expected bits set for our given feature per filter count as given in 3.1. Once all of the bits for a given neuron have been adjusted with randomized rounding, the remaining neurons in the neighborhood of the winning neuron are adjusted. This process is repeated until all the epoch-iteration steps are complete, at which time the neuron adjustment phases ends.

The second stage of training consists of one final pass through the Bloom filter list in it's entirety where each filter is mapped to it's final best matching neuron in the SOM. At this point, the map is considered fully trained.

Several files are generated at the conclusion of the SOM training.

- `<file_name>.som` - SOM configuration file details along with a bit for bit copy of each neuron's feature vector. This file can be used to reload a SOM into memory.
- `<file_name>.best_matches` - Lists every neuron of the SOM and the associated Bloom filters that were matched to them during the final phase of training.
- `<file_name>.records` - For each Bloom filter in each sdbf digest, lists the row and column of the best matching neuron and the best matching score for that neuron for that Bloom filter.
- `<file_name>.sample_count` - For each Bloom filter, the generation number where it was first randomly selected as an input vector for adjusting the SOM neurons.
- `<file_name>.sdbf_to_filter` - A listing of all the digests in a sdbf collection and the Bloom filter index values used during the random selection process.
- `<file_name>.sd_collection` - A by-name listing of each sdbf digest in the collection along with it's Bloom filters represented as 0's and 1's.

Heat maps can be generated from the `<file_name>.best_matches` files to visualize any clustering of Bloom filters within the SOM. This provides a quick check on whether the training cycle was successful or if the configuration file parameters need to be adjusted before continuing to the next experiment.

As discussed in Section 2.6, two files can be considered similar beyond what is expected by chance when sdhash returns a score of 21 or greater.

```
$ sdhash -c sdbf_file_used_in_training reconstructed_sdbf_file
```
(3.2)

3.1.2 Generating sdbf Files from SOM Neurons

After training, the SOM neural network is composed of a large two dimensional array of feature vectors with the same dimensionality as an sdbf digest Bloom filter. Additionally, the SOM will have clustered similar neurons in close spatial proximity. At this point, if we present a sdbf digest to the SOM neural network, it's component Bloom filters can be mapped to the neurons of the SOM that most closely resemble their set bits. For a sdbf digest with a single Bloom filter, this will result in a identifying a single neuron that the Bloom filter most closely resembles. In the case of a sdbf digest with multiple Bloom filters, we will be able to identify an equal number of SOM neurons that most closely resemble each individual Bloom filter.

If we take the collection of neurons that best match the Bloom filters from our sdbf digest, we can use the feature vectors of the neurons as replacement Bloom filters. By combining consecutive neuron feature vectors together as replacement Bloom filters, we can reconstruct an sdbf digest based solely on the neurons in the SOM. If the SOM has been adequately trained, the reconstructed sdbf should retain enough features in common with the original sdbf digest that it can qualitatively demonstrated similar using the tools provided with *sdhash 2.1* for querying 2 files.

A program was developed to take an sdbf digest collection as an input for our SOM. The program presents each individual digest one at time to the SOM, and the component Bloom filters are mapped to the SOM neurons that most closely resemble the filters. After each digest's Bloom filters are completely mapped to the SOM, the feature vectors for each best matching neuron replaces the digest's original Bloom filters. By substituting the feature vectors as Bloom filters, we are able to reconstruct a sdbf digest based on the SOM neural network representation of it. The newly created reconstructed sdbf digest is then saved for later analysis.

3.1.3 Government Document Corpus

The experiments covered in this research make substantial use of the Naval Postgraduate School, Computer Science Department, Document and Media Exploitation Lab government document corpus (govdocs), a collection of over 1 million documents culled from the .gov domain [17].

The collection is arranged into 1000 sub collections, with each collection containing about 1000 individual files. The file types contained in this collection are varied and represent Microsoft Office documents, compressed files, images, htmls, database files, and other document types. For this thesis, six govdocs collections were randomly selected: Five to be used for training

Block Mode	Non-Blocked		4K		8K		16K	
sdbf digest count	983		983		983		983	
sdbf digests with single filter	153		64		125		197	
	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev
Filters per sdbf digest	28.51	91.54	74.60	256.35	37.57	128.17	19.06	64.08
Features per filter	159.08	15.64	66.81	14.27	132.10	29.88	176.59	37.06
Features per single filter	83.40	45.88	40.55	17.33	70.74	37.20	110.70	61.70
Features in tail filter	97.98	44.61	34.12	21.51	59.80	40.12	102.78	65.88

Table 3.1: Collection 017 feature means and standard deviations for various block modes

individual SOMs and initial similarity testing (017, 150, 370, 597, and 721), and collection 795 for final testing with a collection not seen during any SOMs training cycle.

3.1.4 SDHASH-2.1

At the time of the initial framework design, the most current release of `sdhash` was downloaded from `http://roussev.net/sdhash/sdhash.html`. From this source code, a few minor modifications were made to it's core files. Specifically, functions have been added to to access the individual bits of a Bloom filter in order to set their value or return their value to the SOM training program discussed below.

Once compiled and installed, using `sdhash` is straightforward.

- `sdhash -f file_list > output_file` - create a collection of sdbf digests in a single .sdbf digest. A digest is created for the collection from every file name specified in `file_list`.
- `sdhash -b<number> -f file_list > output_file` - create a collection of sdbf digests in a single .sdbf digest exactly as the `-f` option above, except the block mode of the digests is set from non-block mode to <number> KB block size (4, 8, or 16).
- `sdhash -c sdbf_file_1 sdbf_file_2` - compare the similarity between sdbf digest 1 and sdbf digest 2. The score is an integer between 0 and 100.

For each file in all of the government document collections selected in 3.1.3, sdbf digests were created in standard (non-block) mode as well as with block mode alignment sizes of 4K, 8K, and 16K.

After creating several thousand sdbf digests in various block modes, means and standard deviations were calculated against several features of the sdbf digests. Table 3.1 lists statistical data for document collection 017. Through comparison across the four separate sdbf block sizes (standard non-block mode, 4K, 8K, and 16K block alignment), we decided to use one of the

Collection	Filters per sdbf digest		Features per filter		Features per single filter		Features in tail filter	
	Mean	StdDev	Mean	StdDev	Mean	StdDev	Mean	StdDev
017	41.90	136.39	134.08	26.96	135.14	11.82	117.18	42.85
150	60.98	165.61	132.65	32.92	134.47	17.39	118.17	43.73
370	66.76	249.54	132.95	32.21	135.60	14.43	110.80	46.66
597	111.12	422.96	151.21	35.44	126.58	24.31	108.77	46.25
721	48.59	159.27	126.41	39.48	137.98	20.03	111.12	48.42

Table 3.2: Document collections in 8K block mode means and standard deviations after Bloom filter modifications

block modes in order to provide a direct mapping of data in the original file to the Bloom filter. Among the block modes, 4K was removed as an option due to it's large standard deviation in filters per sdbf digest. Between the 8K and 16K block modes, we decided to use 8K block mode due to the larger mean filters per sdbf digest. The sdbf digests with a larger number of Bloom filters provide will provide greater diversity for the individual neurons to be trained with, and 8K has on average more Bloom filters than 16K.

During sdbf digest creation (Section 2.6) each Bloom filter has a preset maximum number of features that can be inserted into it. As this maximum number of features is reached, no additional feature insertions are possible. A new Bloom filter is created for the additional features and the insertion process continues. In Table 3.1, we see that the standard deviation for features increases with single Bloom filter sdbf digests as well as in the last (tail) Bloom filter of each sdbf digest. The increasing variance in features in these two categories could potentially impact the performance of the SOM during training. Bloom filters composed of few features will have a bias towards any neuron whose feature vector has the fewest bits randomly set during initialization. As a result, we decided to remove any sdbf digest composed of only a single Bloom filter from our training and testing collections. Additionally, we stripped the tail Bloom filters from the remaining sdbf digests. This left us with sdbf digests whose Bloom filters displayed a more consistent feature per Bloom filter ratio with a smaller standard deviation. Table 3.2 shows the updated statistical data for the document collection in 8K block mode after removing single Bloom filter digests and stripping the tail Bloom filters from the sdbf digests.

Given the mean features per filter, we can then determine the expected number of bits set in any given Bloom filter with equation (3.1)

For collection 017, after removing all single Bloom filter sdbf digests and dropping the last Bloom filter on the remaining files, our expected bits set is 571.84. We can then tune the SOM neural network so that during neuron initialization, the expected number of bits set per Bloom filter is the same as our expected number of bits set for a particular document collection as discussed in 3.1.1.

3.2 Experiments

After we create a SOM neural network trained on sdbf digests as well as a tool for constructing a sdbf digest from the feature vectors of neurons, our experiments will attempt to determine how similar original sdbf digests are to the constructed sdbf digests. Using a SOM for digital forensic triage is only feasible if we are able to show that the constructed digests are in fact similar to their original digests. If the two digests have no similarity, then the SOM as trained is not useful for clustering unknown files with files of interest to the examiner.

In chapter 4 we present the results of constructing a sdbf digests from the neurons of the SOM neural network. Specifically, our experiment will use document collections used during training as well as the previously unseen document collection 795 as input to our SOM neural network. We will then validate whether similarity exist between a sdbf digest and a sdbf digest constructed from the neuron feature vectors of a SOM neural network. If the two files are similar, then the use of a SOM neural network to cluster files from newly acquired media collections during a digital forensic triage is feasible.

3.2.1 Experiment 1: Trained Document Collection Similarity

After training the SOM with a specific document collection, the first experiment will test for similarity between the sdbf digests in the document collection and the sdbf digests constructed from the neurons of the SOM. This experiment will be conducted against 5 trained SOMs. Each SOM will be trained on a different document collection. If the SOM has been properly trained, the resulting constructed sdbf digests should be very similar to the original sdbf digests on which the SOM was trained. This similarity should be quantifiable using the included similarity querying in `sdhash`.

For each digest in the collection, the following steps will be performed on a trained SOM:

- The training set of sdbf digests will be used as input to the SOM, and for each digest's Bloom filters, the best matching neuron in the SOM will be identified.
- The feature vector for each neuron identified as a best matching neuron will replace the corresponding Bloom filter from the sdbf digest.

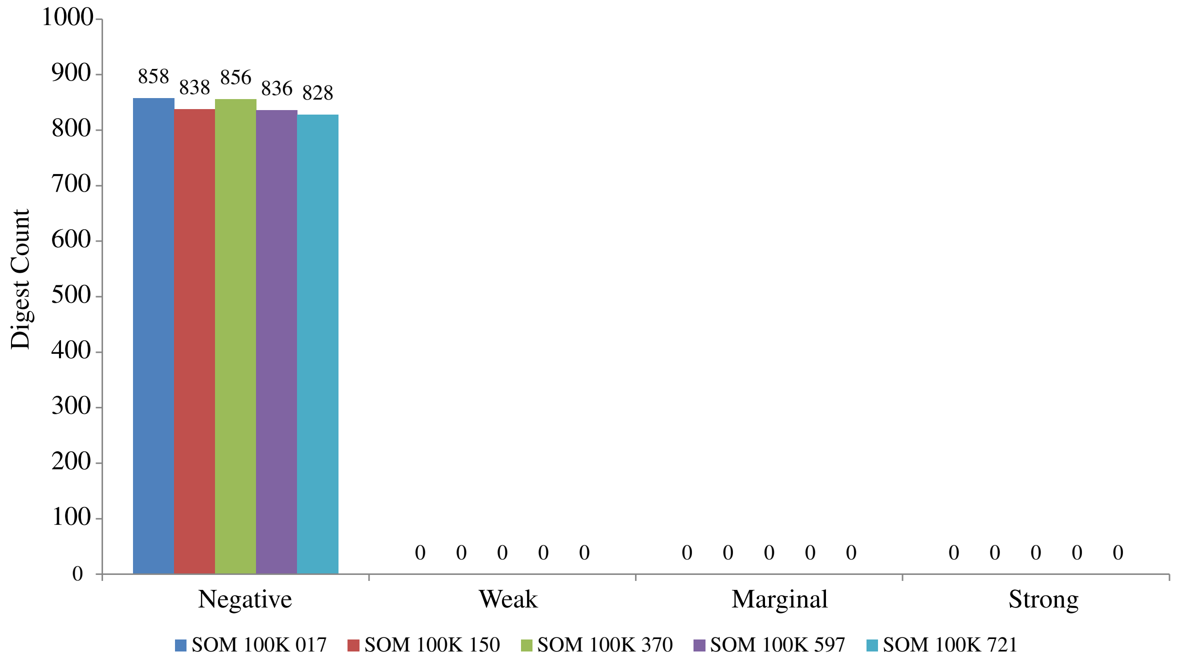


Figure 3.1: Counts of digests binned according to similarity between original and generated digests with no training cycles. Results presented for the 017 govdocs sub-corpora with an untrained SOM.

- The newly constructed sdbf digest, generated from the SOM's feature vectors, will be compared to the original sdbf digest using `sdhash` to find their similarity score.

After all the scores have been calculated, we can then analyze the similarity between the original sdbf digests and the generated sdbf digests. Scores greater than 21 indicate that similarity does exist between the two files. In this case, the trained SOM was successful in clustering similar input vectors from sdbf digests and in recreating a representation of the original file based on a series of best matching neurons.

Cases where the scores are between 11 and 20 indicate only a marginal similarity. This typically is measured similarity in embedded objects in package container type files [2] and not of significant interest in providing contextual information to a forensics examiner. As such, marginal similarity scores will be treated as negative similarity (a score of 0).

Cases where the scores are between 1 and 10 indicate weak similarity prone to false positives. These will also be treated as negative similarity.

As a baseline to compare the experiments with, a SOM was initialized but not trained. SOM generated sdbf digests were generated from mapping collection 017 to the untrained SOM neural network. The generated sdbf digests were scored for similarity against the original sdbf digests of document collection 017. As seen in the histogram of similarity scores (3.1), the untrained SOM neural network resulted in no similarity between any original sdbf digests and neuron feature vector generated sdbf digests.

3.2.2 Experiment 2: Unseen Document Collection Similarity

The second experiment will be carried out in the same fashion as the first experiment. The only difference will be that instead of using the same collection of sdbf digests that a given SOM was trained with, a previously unseen collection (795) will be used instead with each of the 5 trained SOMs. For each file in this new collection, the following steps will be performed:

- The new set of sdbf digests will be used as input to the SOM, and for each digest's Bloom filters, the best matching neuron in the SOM will be identified.
- The feature vector for each neuron identified as a best matching neuron will replace the corresponding Bloom filter from the sdbf digest.
- The newly generated sdbf digest, created from the SOM's feature vectors, will be compared to the original sdbf digest from the new collection from which it originated, using `sdhash` to find their similarity score.

After all the scores have been calculated, we will examine the similarity between the original sdbf digests from the previously unseen collection 795 and the generated sdbf digests. If the SOM has been properly trained, this thesis will show that the resulting collection 795 constructed sdbf digests should be very similar to the original collection 795 sdbf digests, even through the SOM was not trained on this collection . This similarity should be quantifiable using the included similarity querying in `sdhash`. Any score of 21 or greater will show that a SOM can be used to cluster and categorize data during a digital forensic triage.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4: Experimental Results and Analysis

In this chapter we present and interpret the results of the experiments discussed in Chapter 3. The experiments determine the similarity between sdbf digests and corresponding generated digests created from Bloom filter mappings of SOM neuron feature vectors with the hope that as training increased, the similarity would also increase indicating that the SOM was capturing the similarity relationships present in the SDBF digests. In 3.2.1 we discussed how an untrained SOM produces sdbf digests with zero similarity to original sdbf digests. If we can train a SOM and produce similar sdbf digests, we will be able to show that the SOMs are capable of clustering data during a digital forensics triage.

4.1 SOM Training

We initialized and trained a total of twelve SOMs with the document collections 017, 150, 370, 597, and 721. Five SOMs were trained with 100,000 total training steps. Another five SOMs were trained with 1 million training steps. The remaining two SOMs were trained with 2 million and 3 million training steps each. Table 4.1 lists the parameters used for training each of the SOMs. Only document collection 017 was used for training the 2 million and 3 million training step SOMs. Each collection was processed with sdhash-2.1 in 8K block mode. Any digests with only a single Bloom filter were omitted from the collection as these filters typically have too few features set (See Table 3.1. Likewise, the last Bloom filters from the remaining digests in the collection were removed.

After training all the SOMs, it was observed that each of the SOMs exhibited occurrences of a large number of Bloom filters mapped to a single neuron. These single neuron's contained Bloom filters from between 35.78% and 76.25% of all sdbf digests in each collection. Table 4.2 lists these max Bloom filter neurons and the total number of unique sdbf digests mapped to the neurons. Upon further examination, it was observed that for each given neuron, the Bloom filters associated with it have a high level of dissimilarity between each other. Table 4.3 lists the average Hamming distance ratio for each SOM's max Bloom filter neuron. Since Hamming distance gives us the total number of bits that are different between the neuron's feature vector and the bits set in a Bloom filter, these ratios all show that the average Bloom filter is almost completely dissimilar to the neuron it has been best matched to (Filter to Neuron in Table 4.3

SOM	Training Collection	Total Digests	Size of Map (Row:Col:Depth)	Steps (Epochs:Iterations)	Learning Rate	Sigma Ratio
SOM 100K 017	017	858	150:150:2048	100,000:1	0.9	0.1
SOM 100K 150	150	838	150:150:2048	100,000:1	0.9	0.1
SOM 100K 370	370	856	150:150:2048	100,000:1	0.9	0.1
SOM 100K 597	597	836	150:150:2048	100,000:1	0.9	0.1
SOM 100K 721	721	828	150:150:2048	100,000:1	0.9	0.1
SOM 1M 017	017	858	150:150:2048	1,000,000:1	0.9	0.1
SOM 1M 150	150	838	150:150:2048	1,000,000:1	0.9	0.1
SOM 1M 370	370	856	150:150:2048	1,000,000:1	0.9	0.1
SOM 1M 597	597	836	150:150:2048	1,000,000:1	0.9	0.1
SOM 1M 721	721	828	150:150:2048	1,000,000:1	0.9	0.1
SOM 2M 017	017	858	150:150:2048	2,000,000:1	0.9	0.1
SOM 3M 017	017	858	150:150:2048	3,000,000:1	0.9	0.1

Table 4.1: Configuration file settings and Self-Organizing Map Parameters for the 100K, 1M, 2M, and 3M training cycle SOMs

SOM	Total Total	Total BFs	Max BFs in Single Neuron	Unique Digests in MAX BF Neuron	BFs within 2 Neurons
SOM 100K 017	858	35947	2658 (07.4%)	425 (49.5%)	4424 (12.3%)
SOM 100K 150	838	51100	2457 (04.8%)	344 (41.1%)	7470 (14.6%)
SOM 100K 370	856	5714	6058 (10.6%)	492 (57.5%)	9307 (16.3%)
SOM 100K 597	836	92899	4675 (05.0%)	422 (50.5%)	11019 (11.9%)
SOM 100K 721	828	40236	6236 (15.5%)	568 (68.6%)	12846 (31.9%)
SOM 1M 017	858	35947	3564 (09.9%)	485 (56.5%)	5481 (15.2%)
SOM 1M 150	838	51100	12174 (23.8%)	639 (76.3%)	15699 (30.7%)
SOM 1M 370	856	57143	6016 (10.5%)	505 (60.0%)	10669 (18.7%)
SOM 1M 597	836	92899	8006 (08.6%)	491 (58.7%)	29852 (32.1%)
SOM 1M 721	828	40236	4623 (11.5%)	448 (54.1%)	10298 (25.6%)
SOM 2M 017	858	35947	1253 (03.5%)	307 (35.8%)	5538 (15.4%)
SOM 3M 017	858	35947	4014 (11.2%)	505 (58.9%)	6017 (16.7%)

Table 4.2: Maximum number of Bloom filters and unique sdbf digests mapped to a single neurons

and that each Bloom filter best matched to the neuron is also nearly completely dissimilar to every other Bloom filter in the neuron (Inter-Filter in Table 4.3. The expected bits set and actual bits set for each SOM is also listed in Table 4.3. We can see that for each of the neurons, the actual bits set is less than the expected bits set. As a result, these neurons appear to act as a "catch all" neuron within the SOM. Their low bits set create a bias to best matching a large number of Bloom filters to these individual neurons. As there is no relationship between training cycles and max Bloom filters in a single neuron, future exploration of SOMs for this application may benefit from creating multiple SOMs with the same document collection and identical configuration parameters to find mean and standard deviations for these high Bloom filter neurons.

SOM	Filter to Neuron Hamming Ratio	Inter-Filter Hamming Ratio	Expected Bits Set	Actual Bits Set
SOM 100K 017	0.884	0.915	571.83	476
SOM 100K 150	0.904	0.880	566.70	495
SOM 100K 370	0.896	0.928	567.77	475
SOM 100K 597	0.856	0.893	632.33	550
SOM 100K 721	0.893	0.918	543.93	456
SOM 1M 017	0.885	0.911	571.83	459
SOM 1M 150	0.883	0.895	566.70	432
SOM 1M 370	0.899	0.930	567.77	470
SOM 1M 597	0.859	0.886	632.33	514
SOM 1M 721	0.904	0.926	543.93	462
SOM 2M 017	0.880	0.902	571.83	473
SOM 3M 017	0.885	0.907	571.83	449

Table 4.3: Hamming distance ratios and bits set in max Bloom filter neurons for 100K, 1M, 2M, and 3M training cycle SOMs

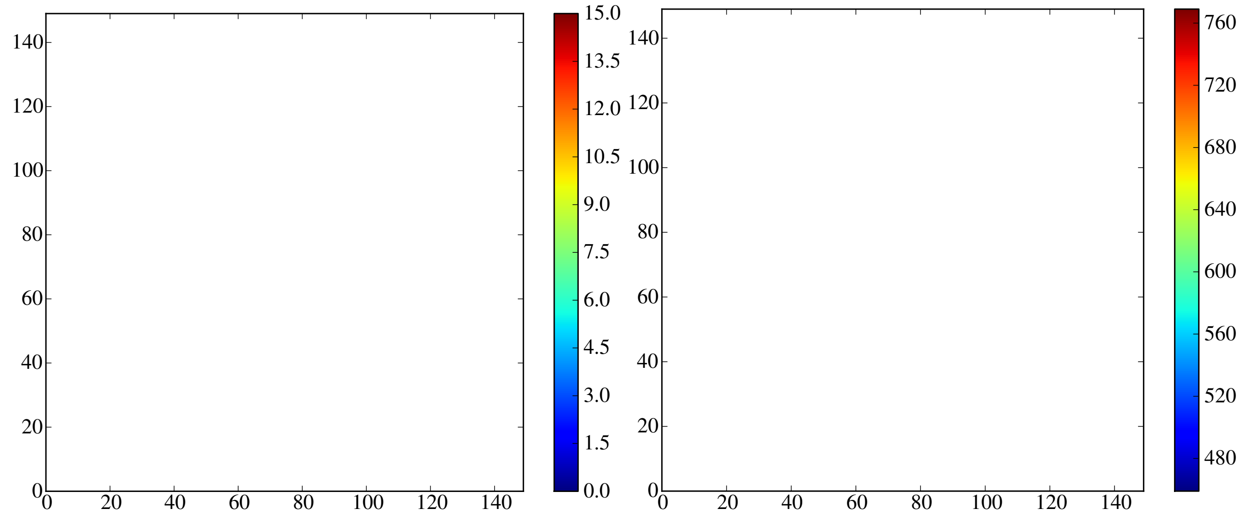


Figure 4.1: Bloom filter heat map (left) and bits set heat map (right) for document collection 017 with 1 million training cycles. Note that the Bloom filter heat map was capped at a max of 15 Bloom filters per neuron to make the image easier to read by diminishing the skewing effects of the max Bloom filter neurons.

Evidence of the bias towards the max Bloom filter neurons can be seen in the heat maps of the Bloom filters per neuron and bits set per neuron (Figure 4.1) for document collection 017 after 1 million training cycles. For ever neuron with a large number of Bloom filters best matched to it, we see a corresponding neuron with a low number of bits set. In general, due to the limited number of trials for each experiment, the outcome of these experiments is suggestive only, and no statistical significance can be attributed to the results below.

4.2 Ability to Regenerate SDBF Digests for Training Documents

After training was completed for each of the SOMs discussed in 3.1.1, the first experiment to reconstruct sdbf digests was performed. The goal of this experiment is to reconstruct the sdbf digests of the document collections we used to train each SOM and qualify how similar the reconstructed sdbf digests are to their original digests. We began with the SOM trained for $100,000$ training cycles on collection 017 (identified as "SOM 100K 017"). For each digest in our training set, we generated a digest using the best-matching neurons from the SOM. We repeated the generation process for document collections 150, 370, 597, and 721, as well as with the SOMs trained for 1 million, 2 million, and 3 million total steps.

For each sdbf digest in the document collection, we identified the best matching neuron of the SOM. Once again, the best matching neuron is the neuron whose feature vectors have the smallest L2 Norm distance from the Bloom filter. Once the best matching neuron was found, the feature vector of that neuron was used to replace the Bloom filter of the sdbf digest. Once all the filters were replaced, we output a SOM generated sdbf digest containing learned approximations of all the digest used for training.

For each generated digest we used `sdhash` to find its similarity with the original sdbf digest. In the following sub-sections, we present the resulting similarities binned as discussed in 2.6:

- Strong Similarity : scores 21 and higher
- Marginal Similarity : scores between 11 and 20.
- Weak Similarity: scores between 1 and 10.
- Negative Similarity: scores equal to 0.

4.2.1 After 100K Training Cycles

Figure 4.2 shows the digest counts binned by similarity to the original digests for each of the govdocs collections. After training for only $100,000$ cycles, we see that our generated digests have become much more similar to their originals as compared to Figure 3.1 where all the digests were clumped into the negative bin. The SOM trained on collection 721 performed the best, having a strong similarity score for 3.26% of the sdbf digests. At the other end of the spectrum, the SOM trained on collection 597 performed the poorest, with 0 digests in the strong category.

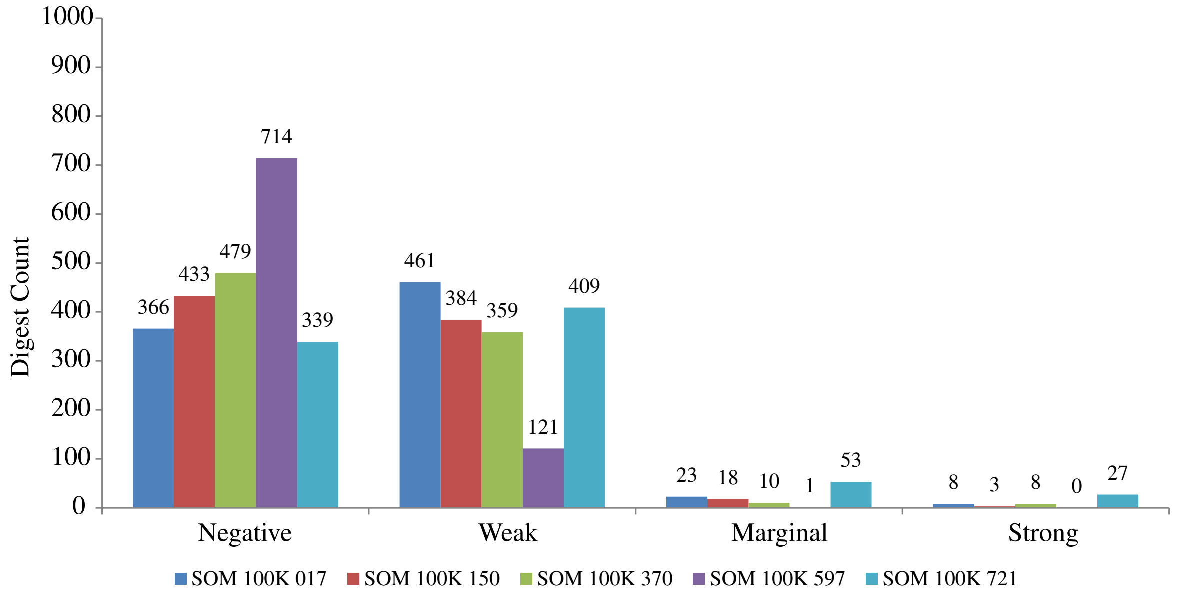


Figure 4.2: Counts of digests binned according to similarity between the original and generated digests after 100,000 training cycles. Results are presented for the 017, 150, 370, 597, and 721 govdocs collections.

4.2.2 After One Million Training Cycles

Similarity scores improved with the one million training cycle digests. Figure 4.3 shows significant increases in strong, marginal, and weak binned similarity counts for each of the govdocs collections. The number of negative similarity scores between the generated digests and the original digests also decreased, especially when compared to the baseline similarity scores in Figure 3.1. The SOM trained on collection 721 once again performed the best, having a strong similarity score for 10.75% of the sdbf digests. The SOM trained on collection 597 once again performed the poorest, with only 0.24% of the digests scoring in the strong category.

4.2.3 2M & 3M SOM Similarity Scores with Training Document Collections

Combining the results of similarity scores between generated digests and original digests for both two million and three million training cycles, we see in Figure 4.4 that the SOMs are returning even more weak, marginal, and strong scores than in the previous training cycles. While this experiment was conducted only with SOMs trained on collection 017, we can see that training cycles does affect the SOMs performance in generating sdbf digests. However, we

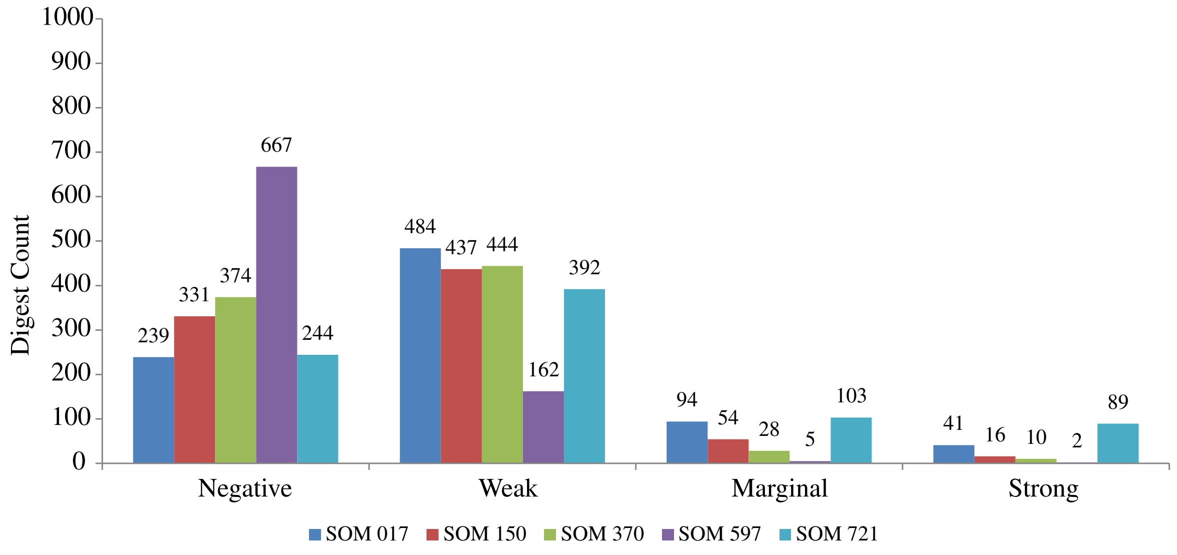


Figure 4.3: Counts of digests binned according to similarity between the original and generated digests after $1,000,000$ training cycles. Results are presented for the 017, 150, 370, 597, and 721 govdocs collections.

note that the three million training cycle experiment resulted in slightly fewer strong similarity scores (5.24%)than the two million training cycle SOM (5.48%). The three million training cycle SOM did have a decrease in negative scores compared to the two million training cycle SOM, as well as an increase in marginal similarity.

4.2.4 Experiment Summary

Figure 4.5 shows the similarity score performance of the document collection 017 generated sdbf digests against the untrained SOM, SOM 100K 017, SOM 1M 017, SOM 2M 017, and SOM 3M 017. As can be seen in the line plot, digests scoring a weak similarity had the largest increase, while the negative scoring digests decreased significantly overall. We can also see that marginal and strong similarity scoring digests enjoyed a small but significant increase as training cycles also increased.

Table 4.4 shows the mean square error (MSE) and mean absolute error (MAE) for document collection 017. For each value, we first calculated the expected similarity score between every permutation of original sdbf digests compared to themselves. This resulted in 734,449 different similarity scores. We then calculated the actual similarity score between every permutation

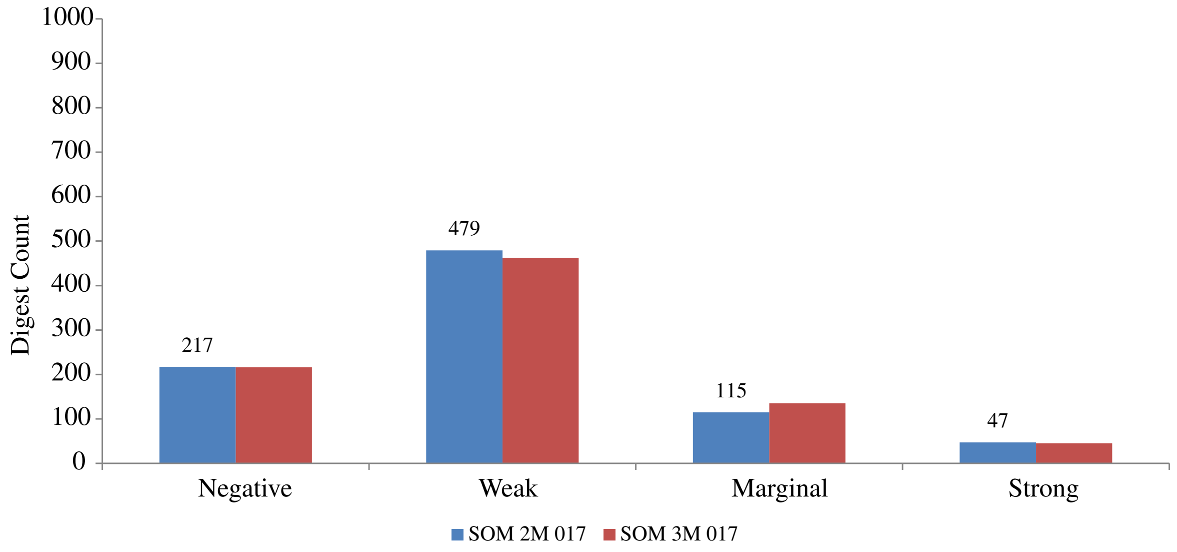


Figure 4.4: Counts of digests binned according to similarity between the original and generated digests after 2,000,000 and 3,000,000 training cycles. Results are presented for the 017 govdocs collection.

Training Cycles	Mean Square Error	Mean Absolute Error	MSE (non-zero scores)	MAE (non-zero scores)
0	12.33	0.162	1209.21	15.841
100K	11.73	0.168	787.59	11.305
1M	11.03	0.165	753.10	11.283
2M	10.81	0.159	759.08	11.161
3M	10.74	0.157	801.42	11.758

Table 4.4: Mean square error and mean absolute error for document collection 017 after 0, 100,000, 1 million, 2 million, and 3 million training cycles.

of the generated sdbf digests compared to the original sdbf digests. The original sdbf digest similarity scores served as our expected values in calculating MSE and MAE, while the original to generated similarity scores were our actual values.

Table 4.5 shows the self similarity MSE and MAE for document collection 017. For each value, we first calculated the self similarity of each original sdbf digest to itself. This yielded one similarity score for each digest, or 858 scores. We then calculated the self similarity between each original digest and the generated digest. The expected values for our MSE and MAE calculations were the original digest similarity scores, and the actual values, once again, were the similarity scores between the original and generated digests.

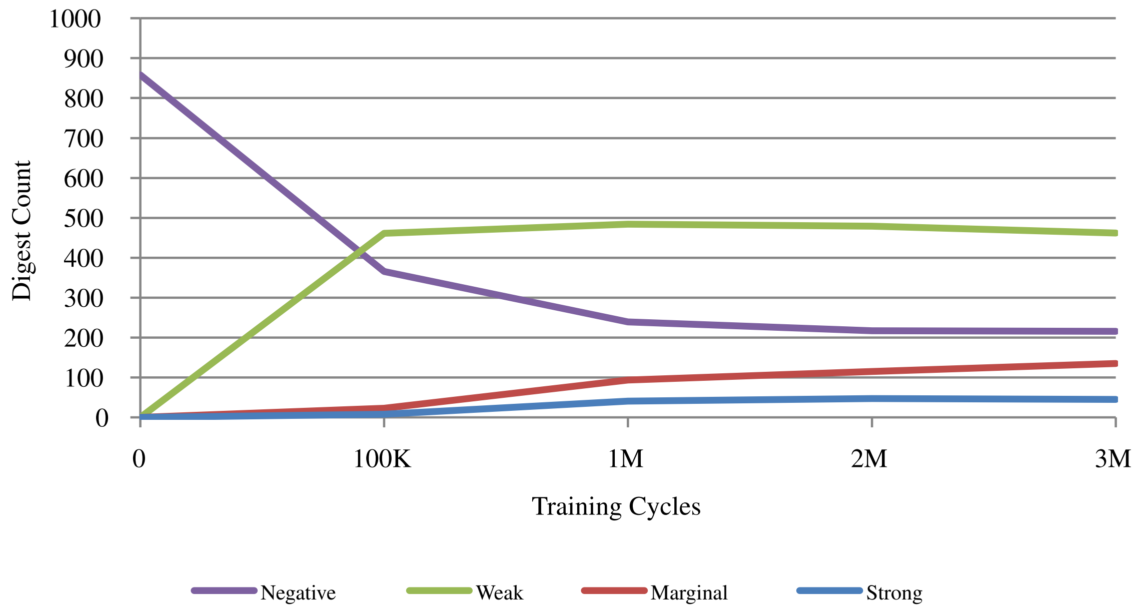


Figure 4.5: Similarity performance for generated digests after 0, 100,000, 1,000,000, 2,000,000 and 3,000,000 training cycles. Results are presented for the 017 document collection.

Training Cycles	Mean Square Error	Mean Absolute Error	MSE (non-zero scores)	MAE (non-zero scores)
0	9128.05	94.467	9138.70	94.578
100K	8633.61	91.782	8643.68	91.889
1M	8015.40	88.312	8024.75	88.415
2M	7933.61	87.822	7942.87	87.924
3M	7874.22	87.499	7883.41	87.601

Table 4.5: Digest self similarity mean square error and mean absolute error for document collection 017 after 0, 100,000, 1 million, 2 million, and 3 million training cycles.

The MSE error compared across the various training cycles clearly shows that as training cycles increase, the similarity between the generated sdbf digests and the original sdbf digests improves. However, taking the MAE into considerations, we also know that very large differences between expected and actual similarity scores occurred. The improved similarity scoring with increased training cycles is also evident in the histogram summaries of all the experiments. This increasing similarity with increased training cycles suggests that a similarity signal is being learned by the SOMs during training.

4.3 Experiments with Untrained Document Collections

In the second set of experiments, we tested sdbf digest generation with document collection 795. None of the SOMs were trained with this collection, so this set of experiments will test whether our SOMs can create strong scoring similarity digests from unknown document collections –a much more realistic test for digital forensic triage. Document collection 795 was processed with `sdhash-2.1` in 8K block mode. Any sdbf digests made of single Bloom filters were omitted from the collection due to low feature element population. Additionally, the last Bloom filter from the remaining sdbf digests in the collection were removed as well.

As before, we began with the 100,000 training cycle SOM trained with document collection 017 (SOM 100K 017). This time, however, we presented it with document collection 795. This experiment was repeated with our other 100K SOMs (150, 370, 597, and 721), as well as with our SOMs trained for 1 million, 2 million, and 3 million total steps.

For each sdbf digest in the document collection 795, each Bloom filter was best matched to a single neuron of the SOM. Once again, the best matching neuron is the neuron whose feature vectors have the smallest L2 Norm distance from the Bloom filter. Once the best matching neuron was found, the feature vector of that neuron was used to replace the Bloom filter of the sdbf digest. Once all the filters were replaced, we output a SOM generated sdbf digest containing learned approximations of all the digest used for training.

For each sdbf digest generated we used `sdhash` to find its similarity with the sdbf digest from which it originated in collecting 795. This process repeated for every digest in the document collection.

4.3.1 100K SOMs Similarity Scores with Untrained Document Collections

Figure 4.6 shows the govdocs collection 795 generated digest counts binned by similarity to the original digests for each of the SOMs. After training for $100,000$ cycles, we see that our generated digests for collection 795 have only a few scores in the weak, marginal, and strong similarity categories. The majority of each collection scores negative similarity, in contrast to the improvement we saw with the trained document collection in Figure 4.2 after an identical number of training cycles. Once again, the SOM training on collection 721 performed the best, having a strong similarity score for 1.04% of the sdbf digests. The SOMs trained on collections 017, 150, and 597 all failed to have a single strong similarity score.

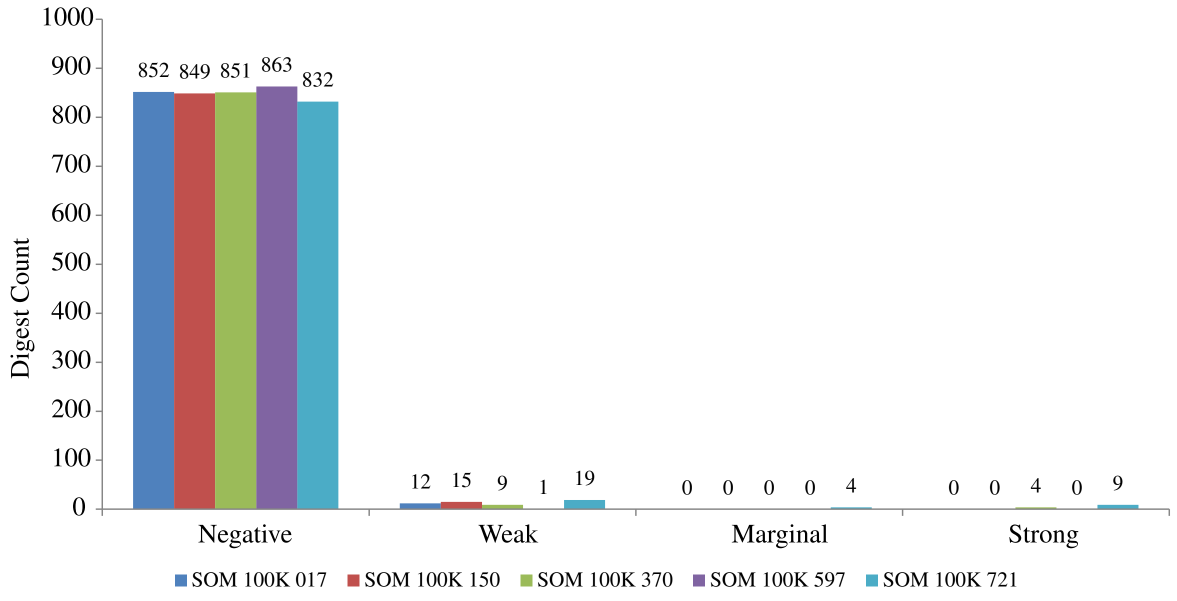


Figure 4.6: Counts of digests binned according to similarity between the original and generated digests after 100,000 training cycles. Results are presented for the 795 govdocs collection.

4.3.2 1M SOMs Similarity Scores with Untrained Document Collections

Increasing the training cycles to one million did not significantly improve our similarity scores, but there were a few minor increases. Figure 4.7 shows this slight improvement over the 100,000 training cycle scores seen in Figure 4.6. While there is improvement, the increase is not on par with the improvement we saw when using the trained document collections as inputs for generating new digests as in Section 4.2. The SOM trained on document collection 721 once again performed the best, having a strong similarity score for 1.97% of the sdbf digests. The SOMs trained on document collection 017, 150, and 597 also failed to have a single strong similarity score again.

4.3.3 2M & 3M SOM Similarity Scores with Untrained Document Collections

Figure 4.8 shows the results of both the two million and three million training cycle SOMs trained on document collection 017 after generating digests for the new document collection 795. The two million training cycle SOM once again performed slightly better than the three million training cycle SOM as we saw in Figure 4.4, but these differences are negligible in the overall poor performance of either SOM to score any significant number of strong similarity

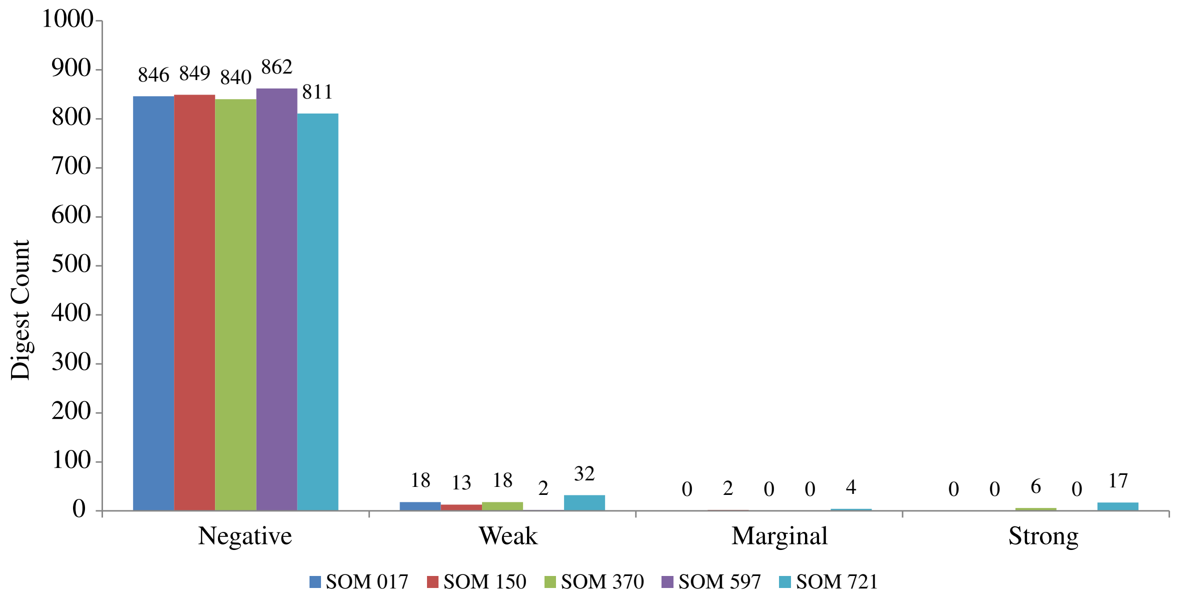


Figure 4.7: Counts of digests binned according to similarity between the original and generated digests after 1,000,000 training cycles. Results are presented for the 795 govdocs collection.

Training Cycles	Mean Square Error	Mean Absolute Error	MSE (non-zero scores)	MAE (non-zero scores)
0	12.43	0.185	1917.96	25.021
100K	12.42	0.183	1799.70	23.826
1M	12.46	0.183	1792.67	23.758
2M	12.38	0.184	1798.10	23.804
3M	12.39	0.186	1792.69	23.758

Table 4.6: Mean square error and mean absolute error for document collection 795 when used with SOMs trained on document collection 017 after 0, 100,000, 1 million, 2 million, and 3 million training cycles.

digests (0.12% each).

4.3.4 Experiment Summary

Figure 4.9 shows the similarity score performance of document collection 795 generated sdbf digests against the untrained SOM, SOM 100K 017, SOM 1M 017, SOM 2M 017, and SOM 3M 017. As can be seen in the line plot, the untrained document collection did not perform nearly as well as the trained document collection experiment. However, we did see similarity scoring improvements, most notably with weak similarity. There was only a slight decrease in negative similarity scores, and even smaller increases in marginal and strong similarity scores.

Table 4.6 shows the mean square error (MSE) and mean absolute error (MAE) for document

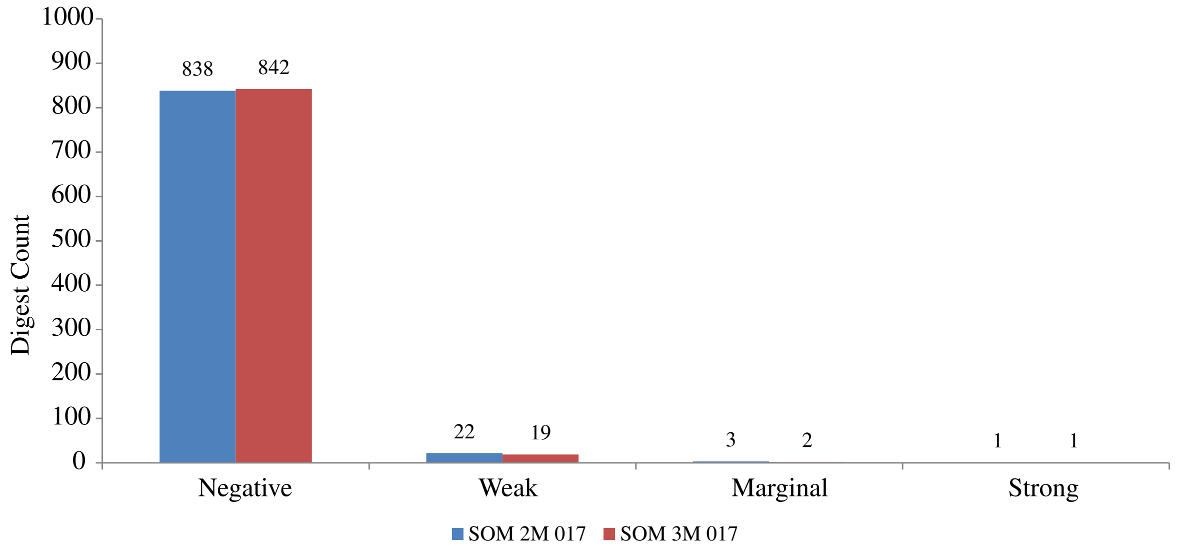


Figure 4.8: Counts of digests binned according to similarity between the original and generated digests after 2,000,000 and 3,000,000 training cycles. Results are presented for the 795 govdocs collection.

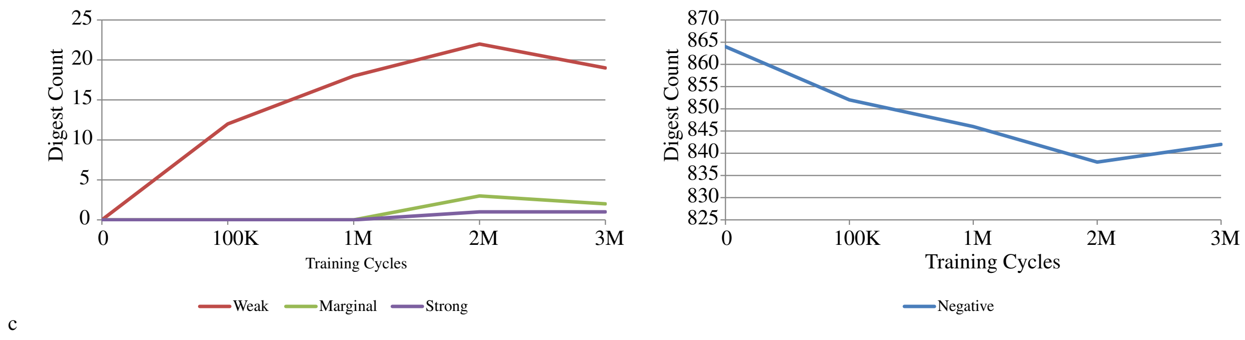


Figure 4.9: Similarity performance for generated digests after 0, 100,000, 1,000,000, 2,000,000 and 3,000,000 training cycles. Results are presented for the 795 document collection.

collection 795. As with the first experiment, we first calculated the expected similarity score between every permutation of original sdbf digests compared to themselves. Next, We calculated the actual similarity score between every permutation of the generated sdbf digests compared to the original sdbf digests. The original sdbf digest similarity scores served as our expected values in calculating MSE and MAE, while the original to generated similarity scores were our actual values.

Table 4.7 shows the self similarity MSE and MAE for document collection 795. We first cal-

Training Cycles	Mean Square Error	Mean Absolute Error	MSE (non-zero scores)	MAE (non-zero scores)
0	9066.96	93.801	9098.56	94.128
100K	9054.97	93.729	9086.52	94.056
1M	9054.70	92.730	9086.25	94.057
2M	9037.23	93.635	9068.72	93.961
3M	9044.92	93.671	9076.43	93.998

Table 4.7: Digest self similarity mean square error and mean absolute error for document collection 795 when used with SOMs trained on document collection 017 after 0, 100,000, 1 million, 2 million, and 3 million training cycles.

SOM	Max BFs in Single Neuron After Training	Document Collection 795 Max Bloom Filters in a Single Neuron	Same Neuron
SOM 100K 017	2658	8722	Yes
SOM 100K 150	2457	6471	Yes
SOM 100K 370	6058	9533	Yes
SOM 100K 597	4675	7457	Yes
SOM 100K 721	6236	10060	Yes
SOM 1M 017	3564	11458	Yes
SOM 1M 150	12174	20837	Yes
SOM 1M 370	6016	10283	Yes
SOM 1M 597	8006	10977	Yes
SOM 1M 721	4623	10060	No
SOM 2M 017	1253	5343	Yes
SOM 3M 017	4014	12840	Yes

Table 4.8: Maximum number of Bloom filters per neuron after training and from document collection 795 best matching. Note: Document collection 795 has a total of 58948 Bloom filters

culated the self similarity of each original sdbf digest to itself. We then calculated the self similarity between each original digest and the generated digest. The expected values for our MSE and MAE calculations were the original digest similarity scores, and the actual values, once again, were the similarity scores between the original and generated digests.

As with the first experiment, the MSE error compared across the various training cycles clearly shows that as training cycles increase, the similarity between the generated sdbf digests and the original sdbf digests improves. However, we did not see the same improvements in scoring as we did with the first experiment. We also know from the MAE that very large differences between expected and actual similarity scores occurred in the experiment. This increasing similarity with increased training cycles suggests that a similarity signal is being learned by the SOMs during training, but this training was not thorough enough for significant similarity scoring with the untrained document collection.

In Section 4.1 we saw that each SOM had a single neuron which collected a large number of best matching Bloom filters from our training documents (Table 4.2). An examination of document collection 795 best matching Bloom filters to neurons provides more insight into how these max Bloom filter neurons bias the best matching process. As can be seen in Table 4.8, in all but one instance, the neuron with the max number of Bloom filters after training was also the neuron with the max number of Bloom filters best matched to it by document collection 795. Additionally, the max number of Bloom filters from document collection 795 in an single neuron is much higher than the max Bloom filters per neuron after training. These neurons clearly influenced the poor performance of similarity scoring with the untrained document collection.

4.4 Summary of Experiments

Overall the SOM neural networks performed poorly but there were some interesting results nonetheless. The experiments using the training documents provided some fairly strong similarity scores, and the majority of digests scored at least weak or higher in these experiments.

The experiments using the document collection 795 did not fair nearly as well, but despite this poor performance, it is worth noting that similarity was found with documents across the 100K, 1M, 2M, and 3M training cycle SOMs.

The poor performance of the various SOM neural networks might be attributable to several factors. For instance, there might not have been enough steps in the training cycle. These experiments used SOMs trained with between 100,000 and 3,000,000 training steps, although as discussed in 2.3, we really should have trained the SOMs with over 11 million training steps. Furthermore, the learning rate or sigma ratio parameters may be improperly tuned. Either of these parameters can dramatically affect the influence each input vector has on altering the SOM neuron feature vectors during training. Too low and the neurons are not influenced enough. Too high and too many neurons are unduly altered, or the influence skews the feature vector too much. This is manifested in the clustering of a large number of Bloom filters into a single neuron as shown in Table 4.2.

The fact that the sdbf digests generated from the SOM neurons did find some similarity in approximately half of all the documents from the trained document collection does suggest that there is validity to this method of data clustering. As discussed in Chapter 2, the baseline similarity scoring for an untrained SOM neural network resulted in zero sdbf digests having any similarity to their SOM neuron generated sdbf digests. However, further exploration of the SOM neural network process with sdbf digests is necessary to improve performance with the untrained document collection to make this method useful in digital forensics triage.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5: Conclusions and Future Work

In Chapter 4 we presented the outcome of our experiments testing SOMs trained on sdbf digests. The results show that a similarity signal is being learned; however, we are not yet able to accurately reproduce sdbf digests from SOM neurons that produce strong scoring similarity across more than a handful of digests. Several factors may mitigate this problem and are discussed in the section on future work below. At this time, the use of a SOM for digital forensic triage is feasible but not fully practical until the sdbf digests generated from the SOM more accurately resemble the original sdbf digests from which they were derived.

5.1 Future Work

This thesis provides an initial foray into the use of self-organizing maps using feature vector representation of `sdhash` Bloom filters for identifying similarity in digital forensics triage. While it has been shown that a self-organizing map used as described in Chapter 3 can recreate a sdbf digest accurately enough to find similarity between it and it's original digest and the similarities improve with training, the performance of this technique must be greatly improved to be useful. In this section, several avenues for further research are presented.

5.1.1 Rational instead of boolean features

When a SOM is initialized, it spends a considerable amount of time training on test data. During this cycle, the neurons of the SOM are adjusted to more closely resemble the test data. The neighborhood influence during this adjustment also produces similarity clustering within the array of neurons. In the experiments conducted in this thesis, the SOM algorithm used feature vectors 2048 bits in length to coincide with the size of the sdbf digest Bloom filters. Furthermore, the values of the feature vectors, like the values of the Bloom filters, were binary - zeroes and ones.

When a neuron was selected for adjustment as either a best matching neuron or as a result of neighborhood influences, each bit of the feature vector was compared to the same positional bit in the input vector Bloom filter. If the two bits matched, no changes were made. However, if the two bits were different, the neuron's bit was changed to a fractional value as calculated in 2.3. However, we then utilized randomized rounding as discussed in 3.1.1 to "flip" the bit to a one or zero centered around a calculated expected number of bits set per Bloom filter. As a result, since the SOM neural network uses a random decision to change a bit or not, it is entirely possible given the large number of training cycles that a neuron's feature vector may not change at all at any given step of the training cycle.

An alternative method would be to replace the binary values of the neuron's feature vectors with rational number values. Instead of using a random number generator to "flip" a bit at a given value, each bit is adjusted within the range of 0 to 1 to a fractional value. The resulting feature vectors would require at least 32 times more storage as each feature would require a floating point value rather than a single bit. While initial experiments with this approach failed to achieve any convergence with training, more investigation with alternate distance metrics may yield better training.

5.1.2 Optimal SOM configuration parameters.

The SOM configuration files offer a straightforward method to alter the training behavior of the SOM. Each of these parameters might be better tuned to improve the overall performance.

- Columns and Rows. The rows and columns of the SOM neuron array were set at 150 each for these experiments. However, neuron arrays in different sizes or shapes are possible. Our decision was based on providing less neurons in the SOM than total Bloom filters in our document collections. However, clustering effects in a larger SOM neuron array were not explored and may provide interesting results where similar neurons clustered together exist as islands surrounded by neurons that do not cluster based on their dissimilarity. Note that increasing the size of the map increases the training time quadratically.
- Epochs and Iterations. The product of epochs and iterations is the total steps implemented in the training cycle. As discussed in 2.3, the number of steps should around 500 times the total number of neurons in the SOM. With our SOM using 150x150 columns and rows, we would ideally want our total steps to be over eleven million. This proved problematic in that a training cycle with this many steps takes a long time to finish. As a result, we ran our experiments with 100,000, 1 million, 2 million, and 3 million total training steps. Providing a training cycle on order of the recommended value may improve the feature vector adjustment and neuron clustering of the SOM to provide a better SOM neural network for the experiments in sdbf digest generation.
- Learning Rate and Sigma Ratio. The SOMs created during the course of this thesis all shared the same learning rate (0.9) and sigma ratio (0.1) values. These values were deter-

mined based on experimentation and code analysis. Given more time, optimal values may be discovered through exhaustive grid searching to produce better training cycle results.

- Parameterless SOMs. An alternative to an exhaustive grid search to optimize learning rate and sigma ratio in our configuration is the exploration of parameterless SOMs [18]. Instead of setting a learning rate and using a decreasing neighborhood size for neuron adjustment, the parameterless SOM adjusts its neurons based on the extent to which the input vectors match the feature vectors of the SOM neurons. This allows for initially coarse adjustment of the SOM with a gradual fine tuning of neuron feature vectors as the SOM adjusts to the data it is being trained on.
- Distance Metric. The SOMs presented in this work used the traditional l2 norm (euclidean distance) for training. A consequence of this is that sparse filters were attracted to sparse neurons resulting in what we call "black holes" in which many filters are trapped. An alternate metric to test is called the Hamming ratio which is the number of features `true` in both vectors normalized by the number of features that are `true` in either vector.

5.1.3 Hierarchical SOMS

As noted in Chapter 4, all of our trained SOMs had what we call "black hole" neurons that had large percentages of the digest filters as best-matches. An alternative/parallel appraoch to using better distance metrics is to use a hierarchical SOM [19] approach in which smaller SOMs are attached under the "black hole" neurons. The smaller SOMs would be initialized so their expected number of set bits matched the neurons from the neighborhoos of the "black hole."

5.1.4 Training Parallelization

As noted above, executing anything but a trivial amount of steps during the training cycle can take a long time. Multithreading the self-organizing map engine would not make it work better, but would allow more rapid testing of alternate parameterizations. Block mode `sdhash` is already configured to work with parallelization, so the component Bloom filters can be created and processed in this fashion. There are two ready opportunities for parallelism in the current implementation. First, in the selection of the best-matching neuron, and second, in the adaptation step for each neuron in the map. Due to our use of randomized rounding in adaptation, we need to be careful to maintain good pseudo-random number generation across threads. A challenge with parallelizing at this level is that each thread only has a relatively small amount of work making it hard to overcome the costs of thread startup and teardown. Courser parallelism across training events is conceptually difficult in the classic SOM framework as potentially all the neurons in the map are adapted in each training event. If the best matching neurons are far enough apart in the map, the training events are effectively independent and the training could run in parallel. As map sizes increase to support training on larger document collections, parallelism becomes mandatory yet also easier as the amount of work in each thread increases while cost of thread startup and teardown remains constant. Further literature review on techniques for parallelizing SOMs is also warranted.

5.1.5 Block Modes

As discussed in 3.1.4, we implemented `sdhash 2.1` in 8K block mode. Using 4K block mode would result in a greater number of Bloom filters per sdbf digest, providing more opportunities for finer similarity comparison at the cost of increasing the time necessary to both train the SOM and map out a given sdbf digest to SOM neurons. On the other hand, 16K block mode feature and filter properties more closely resemble non-block mode digests, and may be worth exploring due to increased performance time (fewer Bloom filters) while still providing a direct mapping of binary information in a file to a specific Bloom filter, an attribute not available in non-block mode.

5.2 Conclusions and Discussions

This thesis addressed the challenge of leveraging the clustering capabilities of self-organizing maps with documents that are only available as `sdhash` digests. In Chapter 3 we presented an adaptation of the SOM that uses a Bloom filter directly as its neuron type giving dramatically reduced storage requirements and increased processing speed. Furthermore, we developed an testing method for analyzing the accuracy of the SOM representation by regenerating documents' `sdhash` digests using only the best-matching coordinates in the SOM.

Beginning with the baseline of a initialized, but untrained, self-organizing map, we note that there existed no similarity between original sdbf digests and sdbf digests created from the neuron feature vectors. Given this universal failure to score similarity with an untrained SOM, our experiments only have to find *any* similarity to show that the technique does work.

Beginning with five randomly selected document collections of approximately 1,000 documents each, we converted them all into sdbf digests through the use of `sdhash`. Next we omitted all sdbf digests composed of only a single Bloom filter as well as stripping out the last Bloom filter from the remaining sdbf digests to decrease the standard deviation of features per Bloom filter. This was done to ensure the Bloom filters used for training and in the experiments were well populated with a full set of features set by a maximum average number of feature elements from the original documents. Each of the five collections were used in training SOMs with training cycles ranging from 100,000 to 3 million steps.

Following training, we generated sdbf digests based on the same documents used to train each SOM. Instead of using the document collections to adjust and cluster the SOM neurons as we did during the training phase, they were instead used simply to find the best matching sequence of neurons in the SOM for each sdbf digest's Bloom filters. From this sequence of neurons, the feature vector of each individual neuron was used to replace the corresponding Bloom filter in the sdbf digest. After every Bloom filter of a given sdbf digest was replaced, we were able to test the similarity between the original sdbf digest and its SOM generated version. We noted that as training cycles increase, the fidelity of the generated digests improved.

The second phase of our experiments used a new document collection not used in the training cycle. Once again, the new document collection was used to find the sequence of SOM neurons most similar to their Bloom filters and new sdbf digests were generated for testing similarity.

While the first phase of experiments were more successful than the second phase in returning similarity between the original and SOM neuron generated sdbf digests, compared to the baseline results of zero similarity, this technique does show promise as an effective tool for clustering binary data during a forensic triage process. The low overall performance of these experiments may be due to SOM configuration parameters that might be better tuned through further experimentation and analysis as discussed previously.

The work presented in this thesis is only the very first steps in applying self-organizing maps to work with `sdhash` digests. If the challenges can be overcome through future work to provide faster training with better clustering, then future work in developing triage tools using the approach can begin. While high-risk, the potential to leverage the proven capabilities of self-organizing maps with the fifty-fold data reduction from `sdhash` digests can be a forensic game-changer.

THIS PAGE INTENTIONALLY LEFT BLANK

REFERENCES

[1] IBM, “Ibm 3390 direct access storage device,” http://www-03.ibm.com/ibm/history/exhibits/storage/storage_3390.html, [Online; accessed 08-December-2012].

[2] V. Roussev and C. Quates, “Content triage with similarity digests: The m57 case study,” *Digital Investigation*, vol. 9, Supplement, no. 0, pp. S60 – S68, 2012, the Proceedings of the Twelfth Annual DFRWS Conference. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287612000370

[3] S. Garfinkel, “Digital forensics research: The next 10 years,” *Digital Investigation*, vol. 7, Supplement, no. 0, pp. S64 – S73, 2010, the Proceedings of the Tenth Annual DFRWS Conference. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287610000368

[4] M. Rogers, J. Goldman, R. Mislan, T. Wedge, and S. Debrota, “Computer forensics field triage process model,” in *The Proceedings of the AFSL 2006 Conference on Digital Forensics, Secuirty and Law*, 2006, pp. 27–40.

[5] S. Garfinkel, “Digital media triage with bulk data analysis and bulk_extractor,” Sep 2011, pre-print submission to Elsevier.

[6] T. Kohonen, “Self-organized formation of topologically correct feature maps,” *Biological Cybernetics*, vol. 43, pp. 59–69, 1982, 10.1007/BF00337288. [Online]. Available: http://dx.doi.org/10.1007/BF00337288

[7] A. Jain, M. Murty, and P. Flynn, “Data clustering: A review,” *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, Sep 1999. [Online]. Available: http://doi.acm.org/10.1145/331499.331504

[8] T. Kohonen, “The self-organizing map,” *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464 –1480, Sep 1990.

[9] T. Kohonen, S. Kaski, K. Lagus, J. Salojarvi, J. Honkela, V. Paatero, and A. Saarela, “Self organization of a massive document collection,” *Neural Networks, IEEE Transactions on*, vol. 11, no. 3, pp. 574 –585, May 2000.

[10] S. Morris, Z. Wu, and G. Yen, “A som mapping technique for visualizing documents in a database,” in *Neural Networks, 2001. Proceedings. IJCNN ’01. International Joint Conference on*, vol. 3, 2001, pp. 1914 –1919 vol.3.

[11] M. Dittenbach, D. Merkl, and A. Rauber, “The growing hierarchical self-organizing map,” in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 6, 2000, pp. 15 –19 vol.6.

[12] G. Salton, A. Wong, and C. Yang, “A vector space model for automatic indexing,” *Commun. ACM*, vol. 18, pp. 613–620, Nov 1975. [Online]. Available: http://doi.acm.org/10.1145/361219.361220

[13] A. Broder and M. Mitzenmacher, “Network applications of bloom filters: A survey,” in *Internet Mathematics*, 2002, pp. 485–509.

[14] V. Roussev, “Data fingerprinting with similarity digests,” in *Advances in Digital Forensics VI*, ser. IFIP Advances in Information and Communication Technology, K.-P. Chow and S. Shenoi, Eds. Springer Boston, 2010, vol. 337, pp. 207–226. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15506-2_15

[15] V. Roussev, “Scalable data correlation,” *Eighth Annual IFIP WG 11.9 International Conference on Digital Forensics*, Jan 2012.

[16] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[17] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, “Bringing science to digital forensics with standardized forensic corpora,” *Digital Investigation*, vol. 6, pp. S2–S11, Sep 2009. [Online]. Available: http://dx.doi.org/10.1016/j.diin.2009.06.016

[18] E. Berglund and J. Sitte, “The parameterless self-organizing map algorithm,” *Neural Networks, IEEE Transactions on*, vol. 17, no. 2, pp. 305 – 316, Mar 2006.

[19] J. Lampinen and E. Oja, “Clustering properties of hierarchical self-organizing maps,” *Journal of Mathematical Imaging and Vision*, vol. 2, no. 2, pp. 261–272, 1992.

[20] J. Vesanto and E. Alhoniemi, “Clustering of the self-organizing map,” *Neural Networks, IEEE Transactions on*, vol. 11, no. 3, pp. 586 –600, May 2000.

[21] D. M. A. Rauber and M. Dittenbach, “The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data,” *Neural Networks, IEEE Transactions on*, vol. 13, no. 6, pp. 1331 – 1341, Nov 2002.

[22] M. Hussin and M. Kamel, “Document clustering using hierarchical somart neural network,” in *Neural Networks, 2003. Proceedings of the International Joint Conference on*, vol. 3, July 2003, pp. 2238 – 2242 vol.3.

[23] H. Yang and C. Lee, “Automatic category generation for text documents by self-organizing maps,” in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 3, 2000, pp. 581 –586 vol.3.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia
2. Dudly Knox Library
   Naval Postgraduate School
   Monterey, California
3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California
4. Directory, Training and Education, MCCDC, Code C46
   Quantico, Virginia
5. Marine Corps Tactical System Support Activity (Attn: Operations Officer)
   Camp Pendleton, California
   Officer students in the Operations Research Program are also required to show:
6. Director, Studies and Analysis Division, MCCDC, Code C45
   Quantico, Virginia
   Officer students in the Space Ops/Space Engineering Program or in the Information Warfare/Information Systems and Operations are also required to show:
7. Head, Information Operations and Space Integration Branch,
   PLI/PP&O/HQMC, Washington, DC